PLANNING IN CONSTRAINT SPACE FOR MULTI-BODY MANIPULATION TASKS

A Thesis
Presented to
The Academic Faculty

by

Can Erdogan

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Robotics

School of Interactive Computing
Georgia Institute of Technology
May 2016

PLANNING IN CONSTRAINT SPACE FOR MULTI-BODY MANIPULATION TASKS

Approved by:

Professor Frank Dellaert,
Committee Chair
School of Interactive Computing
Georgia Institute of Technology

Professor James Kuffner
Robotics Institute
Carnegie Mellon University

Professor Henrik Christensen, Advisor
School of Interactive Computing
Georgia Institute of Technology

Professor Aaron Bobick
School of Engineering and Applied Science
Washington University in St. Louis

Professor Tomás Lozano-Perez
School of Engineering
Massachusetts Institute of Technology

Professor Magnus Egerstedt
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 06 May 2016

To my parents, Nadia and Mithat Erdogan,

to my sister and brother-in-law, Deniz and Emrah Kunt, and baby Idil,

and,

in memory of Professor Mike Stilman.

ACKNOWLEDGEMENTS

I would like to start this acknowledgement with Professor Mike Stilman who was the main reason why I wanted to join the Ph.D. Robotics program in Georgia Institute of Technology in the first place. At the time, in late 2010, his work in humanoid robotics was unique in United States as he was heading one of the few labs with a humanoid robot, Golem Krang, that was designed to perform whole-body manipulation. From our first meeting in March 2011, when we spent the entire interview process talking about humanoids performing martial arts, until the last meeting we had, when he had just submitted our first journal paper on his dream MacGyver project, he was an advisor, a friend and a role model.

Prof. Henrik Christensen has helped me immensely in my last year with the challenging process of evaluating my committee's feedback from my proposal, filling in the holes, writing the thesis and preparing for the defense. Despite his busy schedule, he have always had the time to offer his advice, and I will always cherish his mentorship.

Prof. Frank Dellaert played a vital role in my first year, sharing vital lessons in the entire spectrum of a Ph.D. student's workload. Not only he thought me how to share my work with others by showing how to write technical reports and giving presentations, but also how to even test my thoughts by pair-coding with me, testing ideas in Matlab and writing functional code (and of course, commenting!). His

advisorship in my first year and after Prof. Stilman passed away has been crucial.

It is a unique opportunity for a Ph.D. student to attend his colleagues' Ph.D. proposal and defense talks or even faculty candidate job talks and listen to the faculty ask insightful questions. Prof. Aaron Bobick and Prof. Magnus Egerstedt have been consistently among those who always asked the most interesting questions which either lead to a moment of intellectual deliberation or tough self-reflection for the speaker. Even though it was for a brief few months, having Prof. Bobick as a co-advisor was a challenging but fruitful experience. Their feedback during the development of this thesis has been unquestionably vital to the integrity and completeness of the work.

I would also like to offer my gratitude to my external committee members, Prof. Tomas Lozano-Perez from Massuchusettes Institute of Technology and Prof. James Kuffner from Carnegie Mellon University. Meeting Prof. Lozano-Perez in Karlsruhe, Germany, in 2013, and getting his feedback early in my thesis work has been fundamental in shaping its later chapters. Prof. Kuffner's suggestions on risk in functional structures and design complexity has helped me understand better the practical and theoretical repercussions of my algorithmic choices.

In addition to my committee members, I would like to take this opportunity to acknowledge Prof. Annie Anton, who as the Chair of School of Interactive Computing, has helped me and my friends in Prof. Stilman's group gradually recover from his passing and showed incredible support in the process. Her support in helping us continue our thesis work with minimal interruption and taking the time to mentor us in preparation for our proposal and defense talks were very kind and generous.

One of the reasons that I have been so lucky in my graduate work is the Institute of Robotics and Intelligent Machines (IRIM) center and my colleagues. Starting from my first year, I have had the privilege of collaborating with Yong-Dian Jian, Richard Roberts and Duy Nguyen, who have been both my friends and mentors. Alex Cunnigham and Chris Beall have been always patient with me when I asked

their help on factor graphs and GTSAM. Hanging out in and out of IRIM with Natesh Srinivasa, Ahmad Humayun, Andrew Melim, Rahul Sawhney, Siddharth Choudhray and Tapomayukh Bhattacharjee has always been fun and those times will always be fondly remembered. Writing a paper with Manohar Paluri in my first year on Markov Chain Monte Carlo sampling and its use in 3D perception must have been one of the most challenging and yet fruitful academic experiences of our lives.

Moving onto Prof. Stilman's group and starting to play with Golem Krang and Golem Hubo have led to entirely new interactions with the Humanoids Lab. I have been lucky to have collaborated with Neil Dantam, Tobias Kunz, Martin Levihn, Jon Scholz, Michael Grey, Saul Reynolds-Haertle, Kyle Volle, Stewart Butler, and Peter Vieira. We spent countless hours in the lab with Munzir Zafar trying to get Golem Krang back into shape, which have led to a number of successful demonstrations to news outlets, including Discovery Channel! And of course, Ana Huaman has been a friend (and dare I say academic sister) on whom I can always rely, both for intellectual discussions and friendly chit-chats.

Additionally, I would also like to acknowledge a number of my friends in the other areas of the Georgia Tech world, who have helped me take my mind of my work at times or solve problems by sharing their own expertises in their own academic fields. Dogancan Temel, Emre Yilmaz, Ezgi Karabulut, Melih Turkseven, Ilke Bakir, Didem Pehlivanoglu, Burak Kocuk, Guliz Tokadli, Fatma Karagoz, Kaya Demir and Caglar Caglayan have all been dear friends and confidants. In particular, I would like to thank Akansel Cosgun and Gamze Koseoglu for including me in an adventure and bringing our dog Haze into my life. Co-parenting Haze with them was a special experience and I will always be thankful for their friendship.

The challenge with having old friends in this age is that we have rarely been in the same city more than a few days and long emails/Skype conversations have become our bread and butter. Despite the distances and the time differences, my oldest friends,

Nevin Mutlu, Poyzan Nur Sahiner and Deniz Gungor have always been there for me, showing their support, proof-reading my papers and patiently listening to my gabble on about my research. Simply awesome people!

Finally, I would like to thank my family. Being born into a family of engineers is a unique experience. In my humble opinion, my mother Nadia Erdogan, a faculty member herself in Istanbul Technical University, should have been one of my co-advisors given the number of hours she listened to me go on about the details of my work. My father Mithat Erdogan was a great source of motivation, always chearing me on proudly. My older sister Deniz was always supportive but yet never hesitated to tease me about the bugs in my code. My brother-in-law Emrah Kunt, a roboticist himself, has been a role model, and their daughter, Idil, will always be a source of great happiness in my life!

Table of Contents

List of Tables

List of Figures

SUMMARY

Robots are inherently limited by physical constraints on their link lengths, motor torques, battery power and structural rigidity. To thrive in circumstances that push these limits, such as in search and rescue scenarios, intelligent agents can use the available objects in their environment as tools. Unfortunately, the solution space is combinatorial in the number of available objects and the configuration space of the chosen objects and the robot that uses the structure is high dimensional. To address these challenges, we propose using constraint satisfaction to test the feasibility of candidate structures and adopt symbolic search algorithms to find sufficient designs. The key idea is that the interactions between the components of a structure can be encoded as equality and inequality constraints on the configuration spaces of the respective objects. Furthermore, constraints that are induced by a broadly defined action, such as placing an object on another, can be embedded into logical representations such as Planning Domain Definition Language (PDDL). A classical planning search algorithm thus can reason about which set of constraints to impose on the available objects, iteratively creating a structure that satisfies the task goals and the robot constraints. To demonstrate the effectiveness of this framework, we present both simulation and real robot results with static structures such as ramps, bridges and stairs, and quasi-static structures such as lever-fulcrum simple machines.

Chapter 1

Introduction

1.1 Motivation

The capability to assemble objects in the environment to accomplish a task is crucial for the long-term prosperity of any agent. Especially in adversarial scenarios where the kinematic and dynamic properties of an agent is tested, repurposing everyday objects to utilize them as tools, in order to overcome limitations, would lead to significant advantages. In this thesis, we are interested in a variety of search-and-rescue scenarios, where the agents may not have access to the exact tools that are needed, and subsequently have to use the available resources to complete their missions. Figure 1 below demonstrates several such cases where the participants have to create makeshift bridges over flooded rivers, transport heavy items using simple machine ideas, and rescue people out of collapsed buildings.

The key motivation for this thesis is that in such scenarios, whether the active agents are humans or robots, an autonomous planning system can be utilized to propose "creative" solutions to the problems at hand. The presented algorithms are reminiscent of the "expert systems" in the earlier days of classical planning where

Figure 1: "Experienced humans do not hesitate to use their environment" - Prof. Mike Stilman. A set of examples for humans in need using their environments to overcome their physical limitations.

exhaustive domain knowledge was required as input to the system. In fact, for realistic search-and-rescue planning, we indeed need to be aware of both the agent's properties, such as link lengths, joint torques, power and structural rigidity, as well as the physical properties of the objects in the environment. In contrast with classical planners however, where the configuration space of the agent and the objects would either be individually sampled, either in a uniform or a random fashion, the advocated algorithms in this thesis are based on manipulating whole feasible subspaces of the configuration spaces at once through constraint-based reasoning. We hope to demonstrate to the readers that a constraint satisfaction based approach, embedded in a classical planning framework, can be utilized to propose viable solutions in creating functional structures for search and rescue operations.

1.2 Challenges

Reasoning about how to assemble a set of arbitrary objects to create a useful structure is crucial to achieve tasks that push beyond a robot's physical capabilities. The main challenges that we focus on in this work are as follows:

- **Combinatorial search:** The number of roles for the available objects in the environment for a functional structure is **exponential** in number of objects.
- **Structure configuration space:** In order for a structure to be constructed, the specific poses of its components need to be determined. These poses are expressed in the Euclidean group SE(3) where they denote position and orientations, and thus, for complex structures such as bridges and simple machines, the number of components lead to a search in **high-dimensional** and **continuous** configuration space.
- **Robot limitations:** Every robot has a number of limitations in terms of their motion range, motor torques, battery power and rigidity. Therefore, any structure that is intended to help an agent overcome its physical limitations also have to take them into account. In this body of work, we in particular focus on the kinodynamic limitations of the robots, using the term as defined by [30]. Specifically, we focus on the kinematic limitations on **joint limits and collisions** with the environments, and the dynamic limitations on **motor torques**.
- **Robot configuration space**: In order to take into account robot limitations, one has to reason about the **high-dimensional** and **continuous** configuration spaces of the robot, and that is certainly the case with the humanoid robot such as Golem Krang [122] and Golem Hubo [56].

The main focus of our work is in formulating a framework where a planner can reason about the configuration space of structure components efficiently while also

taking into account the implications of the robot limitations. The key idea is to focus on the poses of the components and the robot at the instances when the robot exerts maximum force into the structure to accomplish a task. For instance, in creating a static structure such as a bridge, where the robot has to walk across the structure, the stability condition is checked only for the two end locations of the bridge. Similarly, for a quasi-static structure, such as a lever-fulcrum system, where the robot pushes on a lever, the initial moment when the robot has to apply the maximum force to actuate the load is considered. The analysis of the structure at a small finite set of moments throughout its use is shown to be sufficient to propose a feasible design.

1.3 Thesis Statement

The constraint representation of functional structures leads us to utilize symbolic search-based planning algorithms where a high-level abstract action, such as stacking two objects, can be used to induce new constraints on the configuration space of the design. Subsequently, a planner can return a skeleton of symbolic actions that describe a design in terms of abstract notions, all the while the constraints associated with these actions limit the design space to only feasible and functional structures. We propose to study the extent of this idea and its applicability to different domains in this thesis work. The following is the main thesis:

> **When robots face tasks that challenge their physical capabilities, they can use constraint satisfaction algorithms within a classical planning framework to create functional structures that incorporate multiple objects.**

1.4 Contributions

The thesis statement has three major concerns: (1) symbolic planners, (2) constraint satisfaction problems (CSPs), (3) functional structures. The goal is to leverage the logical expression of ordering rules in symbolic planning along with the CSP solvers' domain-specific (e.g. Simplex/quadratic programming) and domain-independent (e.g. minimum remaining value) heuristics in creating structures that can help robots accomplish a wider range of tasks. These concepts are better expressed in the form of the following three claims:

- Lazy commitment to continuous variables in the design of functional structures increases planner efficiency and scalability.
- Planning with symbolic actions that involve continuous constraints on contact and positions enables structures that extend the reachable configuration space.
- Imposing robot kinodynamic constraints onto contact positions/forces facilitates structures that extend the force transmission capabilities of robots.

We will use these claims in support of the thesis statement. Our publications are centered around these claims as follows. First, [37] focuses on the problem of commitment in the context of functional structures, and [41] builds upon the use of **lazy commitment** with additional experimental evaluations. Moreover, both pieces of work explore the possibility of creating functional static structures such as stairs, bridges and ramps, using both convex and nonconvex CSP algorithms, to extend the **reachable configuration space** of a robot. Finally, in [38–41], we examine the possibility of increasing the **force transmission capabilities** of robots by designing simple machines and realize these designs in real-world experiments.

In addition to these peer-reviewed publications, we have written a number of technical reports on the physical implementation of the thesis work on the humanoid

robot Golem Krang. These technical reports span the dynamic control of the robot for navigation and mobile manipulation [142, 143], the kinematic modeling of the robot [43] and the evaluation of gravity and drift in the force/torque measurements for whole-body manipulation [42].

1.5 Overview

This thesis is organized as follows. We begin with the literature review in Chapter 2 and present the main framework in Chapter 3. In Chapters 4 and 5, we cover static structures while in Chapter 6, we discuss the incorporation of robot limitations into the design process and simple machines. In Chapter 7, we give an example of autonomous implementation of a lever-fulcrum system by the robot Golem Krang and finally, in Chapter 8, we discuss a domain-independent heuristic based on feasible subspace volumes to increase the efficiency of the constraint-based algorithm.

Chapter 2

Literature Review

Reasoning about using the objects available in the environment is one of the initial motivations in the artificial intelligence community as can be examplified by McCarthy's Monkey and Banana problem in 1963 [104]. In Chapter 1, we have outlined different aspects of the problem, ranging from object properties to motion planning for assembling objects. In this review, we start with the assembly planning problem which focuses on how to motion plan for the construction of a structure given its model. Next, we discuss the task and motion planning literature which shares a common ground with the design problem because the specification of a design contains task-level abstract descriptions and needs to account for the continuous degrees of the robots and objects. Finally, we reflect on the use of constraint satisfaction along with automated planning approaches in different engineering fields such as architecture and chemistry.

2.1 Assembly Planning

We begin with the assembly planning domain which has traditionally focused on the motion generation to assemble the components of a design to create a structure. The motion planning problem is similar to the containment and the separability

problems where in simpler 2D domains, the motion of polyhedral objects has been analyzed [75,98,112]. Kavraki and Latombe are among the first to provide complexity bounds for the assembly partitioning problems showing that the problem is NP-complete [73]. Note that throughout the literature, assembly and disassembly of a structure are deemed to be equivalent since the reversal of the motions to disassemble a design would lead to its construction. Goldwasser and Latombe, later on, extended this work by studying different cost measures, domain-dependent constraints and goals [51].

Most of the work on the domain assumes that a design is provided to the assembly system which later on reasons about the motion planning. Wilson and Latombe are among the first to introduce geometric reasoning about component shapes using non-directional blocking graphs that describe potential collisions among parts [138]. This approach has laid out the foundation for further theoretic and configuration space approaches [58,137], even taking into account the physical tolerations of the parts [62]. Finally, the work of Jones and Wilson on types of constraints used in industrial manufacturing [65] and interactive planning [66] led to the Archimedes 2 system in Sandia National Laboraties [72]. Similar interactive planning and teleoperation schemes have focused on creating assembly description languages similar to PDDL to facilitate the generation of manipulation programs from user inputs [34, 99].

Recently, Xu et al. [140] have proposed the so-called object-oriented templates for complex products where hierarchical actions are introduced as means to facilitate the encoding of interleaved component interactions. We adopt a similar approach in generating lever-fulcrum assembly where a number of constraints between the lever-payload, fulcrum-ground and fulcrum-lever are accounted by a hierarchical action. Moreover, since 2013, a European Union project INTERACT has focused on the autonomous processing of assembly manuals and their augmentation into motion simulation [15]. Finally, Seo et al. [117] has recently focused on multi-robot assembly

where the major challenge is the efficient distribution of the manipulation workload.

This body of work has focused mostly on what we later discuss as the "buildability" problem of a design and has introduced significant concepts such as constraints between parts and partitioning of an assembly. We have extended this approach to autonomously generate designs based on task descriptions that also take into account the physical limitations of the users of the structure. Reasoning about tasks to generate structures with continuous configuration spaces leads us to the task and motion planning domain discussed next.

2.2 Motion Planning

In this section, we present a brief overview of the motion planning literature, starting with the combinatorial motion planning (e.g. roadmaps), then taking into account kinematic limitations and sampling-based motion planning, and finally manipulation planning under uncertainty. First, introduced by Lozano-Perez [96], the *configuration space* of a robot is the space of possible transformations that could be applied to the robot [82].

The overall goal of the **roadmap approaches** is to capture the connectivity of robot's free configuration space in a network of one-dimensional curves [81]. Once a roadmap is generated, the overall plan can be constructed in three parts: first, a subpath from the initial configuration to the start node in the map, then a subpath connected the start and end nodes in the roadmap, and finally a subpath connecting the end node in the roadmap to the goal configuration in the configuration space. There are a number of variants in generating a roadmap that are based on how the nodes are selected in the configuration space. For instance, trapezoidal **cell decomposition** [17] partitions the free configuration space by generating cells where the boundaries of the cells correspond to a change in the constraints that apply to

the robot motion (e.g. ability to move in a specific direction dimension). **Visibility graphs**, on the other hand, propose a shortest-path roadmap where map nodes are generated at the configuration space obstacles' vertices with interior angles greater than π and connected based on whether a collision-free path connects a pair of vertices (i.e. if one vertex is visible from another) [98, 108].

Although roadmap approaches can generate efficient structures for multiple single start-goal queries, their construction takes time for high-dimensional configuration spaces [129]. Another viable option is to treat the robot as a point in configuration space that can be guided by attractive and repulsive potential fields where goal configurations induce a gradient towards them (i.e. attract) while configuration space obstacles push the robot away [76]. Although such **potential field** methods are efficient, because they are essential gradient descent approaches, the robot may get stuck in local minima of the fields.

To address the local minima issues of potential fields and the dimensionality issues of cell decomposition approaches, Kavraki and Latombe [74] have proposed **probabilistic roadmaps** (PRMs) where nodes are generated by sampling the free space of the configuration space and connecting them by checking via a local planner if there exists a collision-free path that respects the holonomic constraints of the robot. One significant drawback of the PRMs is that the learning phase is tailored towards multiple-query planning where the entire static environment is preprocessed without a notion of specific start or goal configuration. Kuffner and LaValle [83] address this issue on single-query planning by generating **rapidly-exploring random trees** (RRTs) by biasing the samples around the start and goal configurations and greedily connect them to the nearest samples in the growing tree. Based on this idea, the RRT literature has expanded into optimal motion planning with RRT* [71], task-constrained motion planning [8, 120], and planning under uncertainty [14].

Although the aforementioned algorithms establish a considerable portion of the

mobile manipulation literature, they are focused on single-task planning where the goal is to get the robot reach a certain configuration space under a number of constraints. The generation of functional structures requires reasoning about how the components of a structure interact with each other and facilitate the robot motion, and thus, we need to expand our discussion to task-level symbolic planning.

2.3 Task and Motion Planning

The ability to reason about task-level problems while carrying out actions in continuous spaces is crucial for robotics. The task and motion planning (TAMP) domain is an example problem where an autonomous agent is expected to reason about a task symbolically while planning the implications of the symbolic actions in its continuous configuration space. A popular approach to this problem uses continual planning where the robot state is monitored by a high-level program which evaluates the changes in the environment lazily and uses cached motion plans [32, 139]. Similarly, Kaelbling and Lozano-Perez propose reasoning online with hierarchical actions where the planner commits to actions early on and replans as necessary [67].

One of the most general TAMP planners is aSyMov [54] which is based on connecting probabilistic roadmaps (PRMs) for the robot and the manipulated objects by sampling the available configuration space. Hauser and Latombe addresses the varying dimensionality of the tasks due to additional constraints by generating multiple PRMs for each feasible space and constructing an aggregate roadmap over the feasible spaces by combining the individual PRMs [59]. Finally, Sucan and Kavraki propose the generation of a task motion multigraphs that that encodes multiple motion plan options and adapts accordingly for a given task description [124].

An important observation in these general approaches is about the flow of information across the symbolic planner and the motion planner. Most approaches regard

motion planning as the lower lever module which simply decides on the feasibility of abstract actions [119]. Choi and Amir [19] have proposed algorithms where symbolic plans can be generated from feasible motion paths.

Another common approach in TAMP domain is using STRIPS-like operators with external validation functions, also called semantic attachments, that address motion planning problems such as collisions [36, 44, 111]. Regardless of the specification of the symbolic search, most TAMP approaches have to commit to the solution of a geometric motion planning problem. When such commitments prove futile as the search progresses, the approaches have to backtrack in the high-dimensional space. Another approach is to propagate constraints throughout the symbolic planning [37, 79]. Lagriffoul et. al propose defining bounds over the possible task-space motions and refining them as additional constraints such as grasping and placement are added to the task description. Recently, Lozano-Perez and Kaelbling proposed an improvement on that idea where the existence of a successful motion plan is treated as another task constraint and the constraint satisfaction module focuses on the most constrained-decisions [97].

The work we present here makes use of multiple key ideas in the TAMP domain such as the augmentation of the STRIP-like operators with feasibility tests and using operators on the continuous space to guide the symbolic search. A major difference is the incorporation of multiple objects in an assembly and the consideration of robot kinodynamic limitations such as joint torques. Moreover, the proposed approach only focuses on the final configurations of objects in an assembly but does not directly refer to the motion plans to position them. Lastly, the framework addresses the continuous object space without the common discretization approach [29, 79, 97] where resolution granularity and efficiency are inversely proportional.

Finally, we would like to make a note on incorporating uncertainty in perception and action into the TAMP problems. Levihn et. al [89] have studied the problem in

the domain of Navigation Among Movable Obstacles (NAMO) where the perception uncertainty affects presumed object categories. To accommodate the long horizon aspect of the NAMO domain, the authors generate two-level MDP representation of the problem where the high-level actions are resolved with the MAXQ algorithm and the near-optimal low level manipulation strategies are determined with Monte-Carlo Tree Search algorithm. In contract, Kaelbling and Lozano-Perez [68] adopt a belief-space approach where hierarchical goal regression is utilized to direct perception actions towards accomplish manipulation tasks while at the same time, planning and execution are interleaved to ensure the subproblems have short horizons and execution errors are handled robustly.

2.4 Constraint Satisfaction Algorithms

In this section, we provide an overview of constraint satisfaction algorithms where the goal is to assign a set of variables specific values from their respective domains such that the constraints on these variables are satisfied. The constraint satisfaction problems (CSPs) can be organized into three categories based on the topology of the variable domains. All the variables might have discrete or continuous domains, or some of them might be discrete, leading to so-called mixed-integer programming problems.

The challenge of finding a value assignment that satisfies all the constraints has been formulated as a backtracking search in the space of partial assignments [133]. In this framework, each state represents an assignment of values to a subset of the variables such that all the relevant constraints are satisfied. A search tree can be generated by connecting two states that change the assignment of a single assignment and a goal state can be searched for by using backtracking in the case of conflicting assignments.

One advantage of using backtracking search is the domain-independent heuristics that can be utilized to direct the search. The **minimum remaining value** heuristic (MRV) biases the search towards assigning values to variables with the fewest permissible values while the **degree heuristic** prefers variables that are involved in the largest number of constraints on other unassigned variables [116]. In addition to heuristics, it is also possible to extend the search to not only assign values to single variables but also alter the available domains of other variables using **forward checking** and **arc consistency** [100].

An important observation in solving general constraint satisfaction problems is the structure and the connectivity of the constraints and the variables affect the search efficiency [23]. Based on this observation, Gottlob et al. [52] have proposed generating a hypertree of a CSP graph such that each node in the tree becomes a cycle of variables in the original problem, whose assignments can be tackled as simpler subproblems, and the tree structure allows simpler consistency checks in the hypertree evaluation. The idea of generating hypertrees have been also extended to probabilistic reasoning domains with Bayes Trees [69] particularly for simultaneous localization and mapping problems.

We have established that discretizing the domain space of the variables is an effective method to reframe a constraint satisfaction problem with challenging nonlinear constraints into a form where backtracking search methods and their accompanying heuristics can be used [97]. However, an appropriate discretization requires domain knowledge to determine the required granularity, and based on the **granularity**, the finite domain space may be too large for classical search algorithms. The advantages and disadvantages of discretization have been analyzed by Floudas and Lin, in the context of scheduling problems for chemical processes [46].

In the rest of this section, we provide an overview of continuous constraint satisfaction algorithms. These problems are identified with continuous domain variables as

well as continuous inequality and equality constraints that bind these variables. Based on the form of the constraints, the simplest class of problems are composed of linear equations where for instance the **Simplex algorithm** with in average polynomial time complexity in the number of variables can be utilized [77].**Conic optimization**, which in fact includes linear programming and is a subfield of convex programming, has been shown to be effective in motion planning for whole-body humanoid behaviors [109].

Although convexity is a desirable property in continuous CSPs, it is not always available and nonconvex optimization methods have to be considered. A general strategy in nonconvex programming is the **branch and bound algorithms** where the variable domains are segregated into smaller problems where gradient descent algorithms can determine the local minimums and select the effective bounds to determine the global minimum [131]. The equality and inequality constraints on the variables are usually treated as cost functions that are to be minimized. In our early experiments, we experimented with a number of global nonlinear optimization such as DIRECT [64] and ISRES [115], along with local nonlinear optimizers COBYLA [113]. The **Levenberg-Marquadt** method with random restarts for initialization has been the main nonconvex optimization algorithm in this thesis [107].

We would also like to make a note on efforts on **analytical methods** for nonlinear programming using algebraic geometry and commutative algebra. Macaulay2 [55] is a software system with a number of tools for polyhedral analysis, Lagrange methods and solving polynomial equations. Despite the theoretical guarantees that analytical methods can provide, their extension to higher-dimensional problems is still a topic for ongoing work [20, 88].

2.5 Planning with Constraint Satisfaction in Different Fields

Finally, recent progress in architectural and furniture assembly problems uses constraint minimization approaches to make design choices [48] and Monte-Carlo sampling methods to design placement of objects in interior decoration [141]. Similarly, work in operations research expresses scheduling problems as temporal constraints within an optimization framework [31]. The key idea is the branch-and-bound approach where the continuous space is partitioned based on discrete choices and then constraint optimization is applied to search for solutions in each subspace [131]. Finally, there has been considerable work in geometric analysis of assemblies and subassembly detection processes in industrial manufacturing [26, 138] along with using motion planning for assembly sequencing by disassembly of proposed structures [84].

Chapter 3

Planning in Constraint Space

To search the configuration space of the assembly components, we propose using a classical planning method where actions represent abstract behaviors such as placing a box on top of another one or pushing on a lever. The planner begins with a set of available objects and searches for a state where the constructed assembly is "useful". Each state throughout the plan represents a partial assembly. The algorithm encodes an assembly using symbolic literals for abstract descriptions, such as On(BoxA, BoxB), and a set of numerical equations that embody the corresponding physical constraints. The planner reasons about the actions with logical operations and constraint satisfaction tests, where an action can be taken if (1) its prerequisite literals are satisfied in the state and (2) there exists an assembly configuration that satisfies the accumulated physical constraints. Once an action is taken, it introduces new symbolic literals and continuous constraints to the state.

3.1 Inputs and Outputs

The domain specification provided by an expert user is crucial in capturing the physical constraints of the proposed assemblies in the symbolic planning framework. Each action $A_i \in \mathbb{A}$, in addition to symbolic prerequisite L_i^p and after-effect L_i^a literals,

requires numerical equations C_i that capture the physics of that behavior. These equations can be equalities or inequalities, and are parameterized with the assembly configurations and the represented robot degree of freedoms. Each equation is a hard constraint that any associated object assembly and robot configuration must satisfy. The actions are instantiated with the available objects $\mathbb{O}$ and their geometric face and edge information (e.g. contact face 3 of object A with face 5 of object B). Therefore, the numerical equations are parameterized by the input object properties $\mathbb{P}_O$ such as the mesh data and the weight. Additionally, the robot properties $\mathbb{P}_R$, such as the joint/torque limits, mass and COM information, may be used. For instance, to overturn a heavy load, both the lever weight and the robot torque limits are needed to determine if sufficient force can be generated. Lastly, the geometric symmetries of the objects, provided by the user, are exploited autonomously to generate the minimal set of faces and edges available for contact.

3.1.1 General Rules for a Functional Assembly and its Design

A set of objects constitute a **functional assembly** if the interactions amongst each other and those with the robot/human help accomplish a given task, as a result of their configurations and physical properties. The robot and/or human that uses the assembly is referred to as a **user agent** while an agent responsible of designing the assembly is referred to as a **designer agent**. We begin with declaring the following self-evident rules:

1. **A task, an assembly and its use:**

   (a) An assembly is either functional or not for a given task and a user.

   (b) A task is the manipulation of an object in the environment (or the user) into a new pose.

(c) A use of an assembly is desired either when alternative motion strategies do not exist (due to obstacles, torque, etc.) or those which exist are less optimal (in terms of energy, time, etc.).

(d) An assembly design that does not take into account the actions of the user is not functional.

(e) The criteria that the assembly has to satisfy to help the user accomplish the task are available to the designer.

2. **Objects:**

(a) The components are solid objects and will remain solid throughout the use (e.g. not ice). They are not deformable (e.g. not sponges).

(b) Before the use of the assembly, all the components are stationary. A component may be positioned in any collision-free configuration in SE(3).

(c) The components can have any geometric (e.g. shape, dimensions) or dynamic (e.g. mass, CoM, friction) property. These properties are available to the designer of the assembly.

3. **User:**

(a) A use of an assembly is a trajectory in the configuration/torque space of an agent where (1) the agent makes physical contact with the assembly, and (2) accomplishes a given task.

(b) The geometric (e.g. link lengths, body shapes), kinematic (e.g. joint angles) and dynamic (e.g. mass, CoM) properties of the user agent(s) are available to the designer.

4. **Interactions between objects:**

(a) The components of an assembly interact with each other either through normal forces (induced by the acceleration of the whole assembly due to gravity or external manipulation), or fixed connections established with welding, screwing, and etc.

5. **Interactions between objects and users:**

   (a) A user agent may move some components (e.g. lever) or the whole assembly (e.g. makeshift hammer) to accomplish a task.

   (b) A user is expected to only interact with the assembly to accomplish the given task and its motion does not contradict the criteria that were used to design the assembly.

   (c) A user does not necessarily assemble the components - another agent may do so.

3.1.2 Simplifying Assumptions

The use of a general functional assembly involves several challenging steps such as precisely modeling the environment and the user agent, encoding the domain knowledge for the designer agent, constructing the assembly, and planning/controlling the user agent's motion. Each of these areas warrant careful study on their own right and ongoing research continuously addresses the issues in depth (see related work).

To refine the scope of the problem on the design process, we make simplifying assumptions on (a) the material properties, (b) the physics that govern the component interactions, (c) the actuation capabilities of the agents, and (d) the domain knowledge available to the agents. Below, we list these assumptions:

1. **Objects:** Each object can be used once in the assembly. Once an agent begins to use the assembly, the *role* of an object can not be changed.

2. **Geometry representation:** The shape of an assembly component or a body part of the user agent is a closed, convex polyhedron, represented as a set of polygons with vertices in $\mathbb{R}^3$.

3. **Contact:**

   (a) Only face or edge contacts are made between two components or a component and the user.

   (b) The face contacts between components have sufficient friction to prevent any relative translation.

   (c) For two objects or an object and the user to interact, there should be contact (e.g. no magnetism).

4. **Execution error:**

   (a) The components or the user agent do not bend or break due to the forces during execution.

   (b) The assembly of the components into the desired poses is possible and has neglible error.

   (c) The user agent can execute its motion with neglible perception and actuation error.

5. **Domain knowledge:**

   (a) There exists a finite set of motion primitives of the user interacting with objects in the environment. Each primitive is a function of a finite set of control parameters, along with the user's kinodynamic properties and the object properties, and when instantiated, outputs a trajectory in the configuration space of the user.

(b) There exists a finite set of object-object and user-object interaction primitives that impose **constraints** on the involved entities' respective configurations and force/torque transmissions, throughout the user agent's motion, and are parameterized by the entities' properties.

3.2 Preliminaries

3.2.1 Constraint Satisfaction Formulation

3.2.1.1 Motivation

The relationship between any two entities i and j of the design, whether they are components, a part of the user or the environment, can be represented either by specifying:

1. a specific relative pose, $T_j^i \in SE(3)$, between the two entities, or
2. an interaction primitive, which in return defines a *subspace of relative poses* $\mathbf{S} \subset SE(3)$, such that any $T \in \mathbf{S}$ satisfies the primitive constraints and can be a viable relative pose, $T_j^i = T$.

We propose that instead of sampling specific relative poses, it is more efficient to only maintain bounds on the space of poses, bounds which are monotonically tightening as an assembly grows in the number of components. In this manner, the design choices can be made without committing to specific object poses, and the final poses are assigned by solving a constraint satisfaction problem, once a set of sufficient bounds are fixed.

3.2.1.2 Terminology and Notation

Definition 1. A **motion primitive** m is a function of the configurations of the objects in the assembly, q_i, the user agent's initial pose, q_0^m, and a set of control

parameters, c^m, and returns a trajectory in the configuration space of the user. Let q^m denote the last pose in that trajectory.

Definition 2. Let M denote the set of motion primitives that the user executes and let q_r denote the user agent's configuration before the use of the assembly. The joint **assembly parameters** with n objects is: $\mathcal{Q} = \bigcup_{i=1}^{n} \{q_i\} \cup \{q_r\} \cup \bigcup_{m \in M} \{q^m, c^m\}$. Similarly, we define the properties for each object and the user agent as: $\mathit{\Pi} = \{\pi_0, \ldots, \pi_{n-1}, \pi_r\}$.

Definition 3. An **interaction primitive** is a tuple of real-valued scalar functions $\mathcal{F}$ and $\mathcal{G}$ where each $f(\mathcal{Q}, \mathit{\Pi}) = 0 \in \mathcal{F}$ is an equality constraint and each $g(\mathcal{Q}, \mathit{\Pi}) \leq 0 \in \mathcal{G}$ is an inequality constraint. There exists at least one $\mathcal{Q}$ that satisfies all the primitive constraints. We denote a primitive with $I = \{\mathcal{F}, \mathcal{G}\}$.

Note that the interaction primitives are a part of the domain knowledge available to the planner (see 1.2.4c) and to come up with a useful feasible design, the planner has to make choices about which *roles* (i.e. interaction primitives) to assign to the components of an assembly.

3.2.1.3 Functionality

Definition 4. There exists a real-valued scalar function $m(\mathcal{Q}) < 0$ that imposes a **functionality constraint** on the design space $\mathcal{Q}$ (See 1.2.4a). A **functional design** is a feasible design that also satisfies the functionality constraint.

3.2.2 Planning Framework

3.2.2.1 Motivation

A feasible design has been defined as the instantiation of continuous configuration variables of the components of an assembly to satisfy a design scheme. Moreover, a

functional design has to generate a plan for how the user interacts with the assembly (See 1.1.1d). Thus, the planner has to:

(a) **choose** the components of the design scheme,

(b) **choose** the interaction primitives that generate object-object bindings,

(c) **choose** the sequence of motion primitives the user agent(s) execute, and

(d) **choose** the interaction primitives that generate bindings between the control parameters of the motion primitives, the initial user pose, and the components.

These choices are limited to a discrete and finite input set (1.2.4b, 1.2.4c). Moreover, the knowledge that informs these choices, specifically the properties of the objects in the environment and the user agent (1.1.2c, 1.1.3b) is available to the planner.

3.2.2.2 Planner setup

We adopt a classical search algorithm where:

- a **state** is a tuple of:
    1. propositional variables
    2. set of equality and inequality constraints, imposed by interaction primitives, and describing a feasible assembly configuration space
- in the **start state**:
    1. all the objects in the environment are available,
    2. the user is on the ground, and
    3. there are not any user-object or object-object constraints yet.
- the **goal state** is characterized by:

1. the goal literal,
2. the existence of a feasible instantiation to the components' and the user's configuration space that satisfies the constraints

- the **actions** are a *set* of interaction primitives along with propositional variables that control the circumstances a primitive can be applied. An action has propositional preconditions and after-effects, and each action imposes the constraints of an interaction primitive. And finally,
- there exists at least one action that has the goal literal as an add-effect and imposes the functionality constraint $m(\mathcal{Q}) < 0$.

3.2.2.3 Connection with constraint satisfaction

At each state, the planner *generates a continuous constraint satisfaction problem* by introducing new constraints on the assembly parameters and accumulating the constraints of its parent state. The existence of a solution to the problem *confirms* that the discrete choices made (1.4.1a-d) are viable to generate an assembly. Note that, the proposed forward search and the additional action ordering requirements presented in Section 1.4.5 introduce an ordering of how the constraints are added to the satisfaction problem. However, the ordering does not affect the existence of a feasible solution. In fact, a search heuristic that induces the most limiting constraints earlier in a plan to determine infeasibility cases to prune preemptively such states would be desirable.

3.2.2.4 Domain definition assumptions

For the search algorithm to output functional designs and user action plans, the action formalism in the domain definition has to satisfy a set of criteria.

(a) If an object is a parameter of an action, then the object is a component of the assembly. If an action introduces a new object o to the assembly, it should indicate it with the predicate $Used(o)$.

(b) Actions can not assign contradictory roles onto objects or reassign objects different roles throughout the use of the assembly (see 1.2.1).

(c) Actions should reflect the addition of new constraints onto the continuous space of assembly parameters with new predicates in the discrete, logical representation of a state.

(d) In addition to generating object-object and object-user bindings between their input parameters, actions may also have to generate bindings between the input entities and the other existent assembly components, by discerning interaction principles from the logical representation of a state (e.g. stacking A on B which is already on C implies generation of A-C bindings). The necessity to generate such additional bindings must be expressed in the action formulations.

(e) The physics model of an assembly (e.g. gravity, welding, etc. - see 1.1.4a) adopted in the constraint formulations of an action should be sufficiently detailed to thrive in real-world testing.

(f) A subset of actions should embody construction strategies such as stacking objects, simple machine principles and etc.

(g) A subset of actions should embody *motion strategies* for the user by adopting specific motion primitives, and imposing constraints between their control parameters and the rest of the assembly parameters. This implies generating constraints between consecutive motion primitives i and j such that the execution of control parameters c^j from the pose q^i leads to the pose q^j. Note that q^m, for any $m \in M$, is a secondary variable that is used as a convenience to

impose constraints on the output of c^m more easily (e.g. foot should be on the ground with COM over support polygon).

(h) Actions adhere to the rules on object properties (1.1.2a) and the stasis of the assembly before use (1.1.2b), as well as the contact assumptions (1.2.3a-c).

3.2.2.5 Expected output

Definition 5. A construction action only imposes constraints between object-object pairs or between objects and the initial robot pose. The symbolic instantiation of a construction action may lead to a new component in the assembly (choice type (a) in 1.4.1). Construction actions always impose constraints on the configuration and force/torque transmissions of the instantiated objects (choice type (b)).

Definition 6. A user action, defined with a specific motion primitive, imposes constraints between objects, the initial user pose and control parameters (choice type (d)).

A plan P is an ordering of construction and user actions. The ordering of the user actions in the plan corresponds to the ordering of the motion primitives the user agent(s) execute (choice type (c)).

Definition 7. Let $\ll_P$ denote an ordering such that for any pair of actions a_i and a_j, the notation $a_i \ll_P a_j$ indicates that a_i precedes a_j in plan P. Let $C(P)$ be the set of construction actions in plan P and let $U(P)$ be the set of user actions. The construction plan P_C based on a plan P is the sequence of construction actions with preserved orderings:

$$\forall a \in P_C, a \in C(P) \quad \text{and} \quad \forall a_i, a_j \in C(P),\ (a_i \ll_P a_j) \implies (a_i \ll_{P_C} a_j)$$

The same construction can be applied to generate a user plan P_U using the set of user actions $U(P)$.

For a plan P to be executable, we also enforce an ordering between construction and user actions such that for any user action $a_u(O) \in U(P)$ that interact with components O in the assembly, any construction action $a_c(O') \in C(P)$ that affects one of the components should come first:

$$\forall a_u(O) \in U(P),\ \ a_c(O') \in C(P),\quad O' \cap O \neq \emptyset \implies a_c(O') \ll_P a_u(O) \tag{1}$$

A construction plan P_C, with instantiated object configurations that satisfy all the interaction primitives in the goal state of P, can be interpreted as a plan to assemble the objects. Note that we make a *strong* assumption on the "assembly problem", that is, there exists a motion plan for the agent responsible of placing the components in the desired configurations (see 1.1.5c and 1.2.3b). A user plan P_U, with instantiated trajectories computed from the primitive motion controllers c^m and the intermediary user poses q^m, for all $m \in M$ in the goal state of P, can be interpreted as an executable plan. Note that the trajectories have to satisfy a set of constraints such as being collision-free and the requirements and strategies for generating such trajectories are discussed next.

Requirements on motion:

The final user motion is the concatenation of the trajectories output by motion primitives with control parameters that satisfy the accumulated constraints in a planner goal state. For the assembly to be functional, each motion primitive trajectory has to satisfy a set of additional constraints:

- configurations should be within joint limits of the user,
- user appendages (links) should be collision-free,
- the torques required to achieve the configurations should be within limits, and

- for balancing robots, the zero moment point (ZMP) should be within the support polygon.

Correctness:

To summarize, in the broadest terms, a plan P is correct, if:

(a) The domain definition respects the assumptions on "one object use", accuracy of physics modeling, contradictary rules, etc. in 1.4.3, and provides construction and interaction strategies,

(b) A sequence of construction actions are generated to place components in stasis before interaction,

(c) A sequence of user interactions with components are generated to accomplish the task, and

(d) Each user interaction follows a trajectory that adheres to the aforementioned motion requirements.

3.2.2.6 Motion trajectories

Computing a feasible trajectory can be challenging, even in non-clustered environments for user agents with relaxed kinodynamic limits, due to the high dimensionality of the user configuration space. In such cases, the domain definition can be simplified by *providing bounds on the control parameters and how they relate to the trajectory generation, such that a planner can assume any output trajectory is already feasible.* In the goal state, of course, a complete motion planner would check the accuracy of the assumptions. Below, we provide three categories with increasing motion planner complexity and delineate their advantages:

1. **Control parameters as expected outputs:** If the user motion can be summarized as the translation of an end-effector or center of mass to another position, the control parameters c^m can be used to express the expected outcome of the motion, with conservative minimum and maximum bounds defined in the action formulation. For instance, instead of solving a ZMP system to take a step, the planner could check if the step sizes are within expected bounds. Advantages/disadvantages:

   - ✓ Efficiency and simplicity: Convex lower and higher bounds on the control parameters
   - × Loss of completeness: For any bound chosen, there may exist motions that transcend it.
   - × Failure to prune: In challenging environments, without collision or torque-limit checking, planner cannot prune states that require infeasible motions.

2. **Control parameters to a fixed motion structure:** If the user motion can be limited to a subset with a lower-dimensional representation, for instance, motion of an end-effector as a sequence of b-splines, then the control parameters c^m of the motion parameter can control the underlying structure. Subsequently, the number of parameters would be limited (assuming a maximum number of concatenated underlying structures), and incorporating collision checks and etc. in the constraint satisfaction incurs minimal efficiency overhead. Advantages/disadvantages:

   - ✓ Incorporating collision and torque checks: Opportunities to prune in challenging environments.
   - × Complexity: The encoding of the constraints in the interaction primitives for the motion incorporates the details of the representation.

× Loss of completeness: For some bound of maximum concatenated structures, there exists a motion that can not be represented.

3. **Control parameters as trajectory samples:** Each control point is a trajectory sample with an assigned index such that consecutive points have a distance constraint in the configuration space of the user. If initial samples can be generated heuristically in relevant poses, then the trajectory would be optimized along with the other assembly parameters. Advantages/disadvantages:

   ✓ Completeness: Any user motion can be generated if the parameter space is well-sampled.

   × Efficiency: The number of assembly parameters is linear with the length of the trajectory and with the dimensionality of the configuration space. Significant scalability issues.

Of the three categories presented, the first option has been successfully tested in stacking and simple machine domains. Among the three options, it postpones the complexity of the motion planner most until the goal state. The second, underlying structure option is theoretically sound, and provides enticing trajectory feasibility check options while still keeping the number of parameters limited for scalability. The third option, while technically providing the full power of motion planning, may be infeasible in practice due top efficiency issues.

3.3 Algorithm

Algorithm 1 takes the domain actions $\mathbb{A}$, the domain objects $\mathbb{O}$, the robot and object properties, $\mathbb{P}_R$ and $\mathbb{P}_O$, the initial state S^0 and the symbolic goal literal l^g, and searches for a feasible assembly. The initial state S^0 is characterized with an empty set of numerical constraints and a set of literals $\{\neg Used(o_i) : \forall o_i \in \mathbb{O}\}$ which

indicates all the objects are available for use. A goal state S^g contains the goal literal input l^g - for instance, for a domain where the robot needs to climb an obstacle K, the literal would be $At(K)$. Additionally, the continuous constraints C^g of the goal state would be satisfiable by an assembly configuration X, which would help the robot accomplish its goal.

Algorithm 1: ConstraintPlanner()

Input: $\mathbb{A}$: domain actions, $\mathbb{O}$: domain objects, $\mathbb{P}_R$: robot properties, $\mathbb{P}_O$: object properties,
S^0: initial state, l^g: goal literal, N_f : number of random restarts for nonlinear feasibility test
Result: X: functional assembly configuration

```
1  stateStack ←createStack(S^0);
2  while state ← stateStack.pop() do
       // Instantiate actions and their constraints based on state and objects/robot
          properties
3      actions ←instantiate(A, state, O, P_O, P_R);
4      foreach A_i in the set actions do
5          if L_i^p ⊂ state.literals then            // Check for action prerequisites.
6              C^new ← state.C ∪ C_i;                // Incorporate new constraints.
7              X ← feasibilityTest(C^new, state, N_f) // Find feasible configurations.
8              if X = ∅ then continue;               // If action not feasible, check next.
9              else
10                 if l_g ⊂ action.L_i^a then return X;   // If feasible and satisfies
                   goal, return.
11                 else                              // If feasible but not goal yet, recurse.
12                     child = {state.literals ∪ action.L_i^a, C^new};
13                     stateStack.push(child);
14                     break;
15
16 return ∅;      // If none of the feasible actions reaches the goal literal, return.
```

The algorithm is structured as a depth-first search in a tree where each state is a partial assembly. The search is organized with a stack data structure (line 1) and for each new state, a set of domain actions are instantiated based on the state, the object properties and the robot limitations (line 3). To apply an action, first its prerequisites are checked (line 5), and then a feasible assembly configuration is searched for the

union of the accumulated state constraints and the new action constraints (line 7). Finally, if the new state contains the goal literal, the configuration X is returned as the desired assembly (line 10); otherwise, the state is pushed onto the stack (line 12).

3.4 Complexity analysis

Algorithm 1 is a depth-first search approach, where at each state, a feasibility test is issued on the accumulated constraints. Therefore, the efficiency depends on the number of states in the search and the feasibility checker. Note that the general depth-first search complexity is $O(b^d)$ where b is the branching factor and d is the worst-case depth for a solution. In the following, we derive these two parameters in terms of the input number of objects n, number of actions types m, and maximum number of parameters to an action k (e.g. $k = 2$ for $PutOn(O_1, O_2)$). Starting with the maximum worst-case depth, observe that each construction action adds a "Used" predicate for the specific object that is newly included in the design. Then at most n construction actions can be taken and if c is the worst-case number of robot interactions actions per object, cn number of use actions can be taken. Then $(c+1)n$ is the maximum length of a symbolic plan. For the branching factor, observe that there would be n^k instantiations for each action. Accounting also for the number of action types, the branching factor of the tree is mn^k. In addition to the tree search, we need to address the complexity of evaluating the feasibility of a state in the tree.

The feasibility test is crucial in assessing the efficiency of the approach. In this work, we utilize two different approaches: simplex algorithm for convex domains and Levenberg-Marquardt minimization approach for nonconvex domains (see Section 6). The simplex method is very efficient in practice and can be considered $O(1)$ complexity given the small number of variables the assemblies require. Thus, the complexity of the framework with convex constraints is $O((mn^k)^{(c+1)n}) \cong O(m^n n^{kn})$.

The combinatorial effect of the number of available objects n can be observed when we compare the computation times between the 3-block example and the bridge example. The simple example takes only 0.02 seconds while the bridge example takes 12.98 seconds, both averaged over 10 trials on a 4.5GHz Intel I7 processor.

Although the analysis of the convex domain is relatively straightforward with the almost negligible overhead of the simplex algorithm as an evaluation test for domains with small number of objects, the same can not be said for nonconvex domains. As we will expand later in Chapters 5 and 6, the challenge is that with a nonconvex domain, the goal becomes finding the global minimum of a cost function and determining if it is valued zero. Although an exhaustive search is not plausible, we adopt a stohastic approach where Levenberg-Marquardt Method is used to minimize the cost from a finite number of randomly picked seed points iteratively.

The challenge is though, in addition to searching the planning tree for meaningful design constructions, finding a sampling a good seed point. Figure 2 below demonstrates the optimization progress during a lever-fulcrum design where the planner first (1) decides where to place the lever and fulcrum without taking into consideration robot's kinodynamic limitations, (2) then the link lengths and robot balancing is considered and (3) robot torque limits are accounted for. The top section displays the number of incremental optimization steps made after each random restart where the algorithm either stops the steps when lack of progress is detected, the maximum number of steps is reached or the local minimum is found. The bottom section displays the cost for each step.

The graphs demonstrate the challenge of sampling to find good seeds (especially for the second action) and even if plausible seeds are found, the optimization may not be fruitful (especially during the evaluation of the first action). To analyze the complexity of the evaluation using Levenberg Marquardt with random restarts, we could estimate the number of random restarts needed to capture a *favorable* seed that

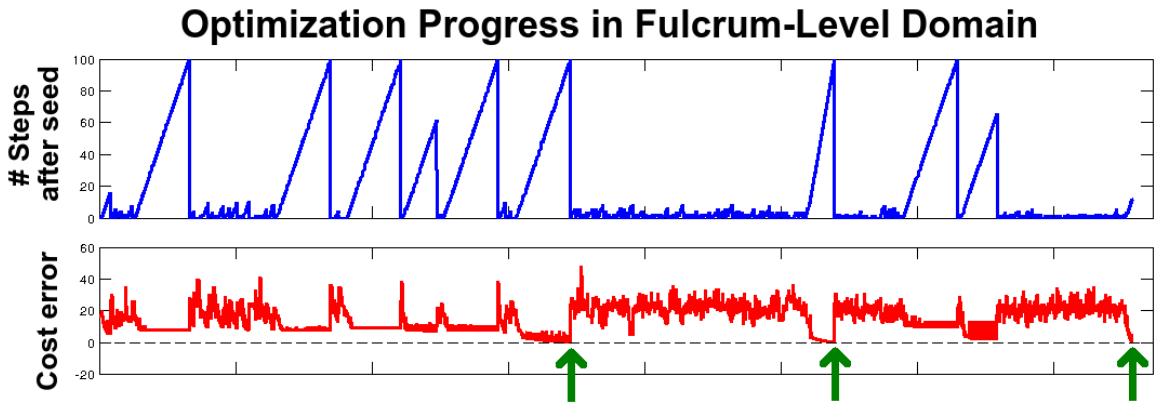


Figure 2: Top: The number of samples generated in Levenberg Marquardt method after each random restart (spikes). Bottom: The cost error induced by the specific samples at the given time.

would lead to the global minimum, if one exists. This can be achieved by introducing new variables for the average number of constraints per action in a final path and making an argument in terms of the volume of the feasible space with respect to the initial bounding box as will be done in Chapter 9. However, such an analysis is unnecessary, since it is the states with failed evaluations that would dominate the computation as can be seen in the top section in Figure 2.

Let $\hat{r}$ be the maximum number of random restarts allowed in the evaluation process and let ϵ be the error tolerance of the Levenberg Marquardt method such that the method stops proceeding if the change in error is less than ϵ. Note that if ϵ is sufficient large, any proposed design by the planner would be deemed feasible by the evaluation test. Ueda and Yamashita offered the first complexity analysis of the Levenberg Marquardt method in 2009 [128] where the upper bound on the number of iterations to get within ϵ error range is determined to be $O(\epsilon^{-2})$. In order to generate an upper bound for $\hat{r}$, we can assume that we perform a discrete sampling of the workspace where if s is the number of samples per dimension and if we assume a 6-DOF configuration space per object, then $\hat{r} \leq 6ns$ - linear in the number of objects. Using this generous upper bound, the complexity for the algorithm can be written as as $O(\epsilon^{-2} m^n n^{kd+1})$ and reflect the dependence of the complexity on the number

of random restarts linearly and the precision of the optimization method by inverse squared. The choice of these specific values are declared in Chapters 5 and 6 with the appropriate experiments but note that the precision variable ϵ is set such that the results would correspond to millimeter level design precision.

3.5 Framing Solution

In this final section, we summarize the properties that would either lead to an ideal or intractable problem to point out the strengths and the weaknesses of our framework. Note that in listing the sources of intractability, we overlook scalability issues and assume we have sufficient computational resources.

3.5.1 Ideal Problem

In the following points, we describe different properties of problems such as the shapes of the available objects to the complexity of the domain actions to give an overview of an ideal problem.

- **Input objects can be clustered based on their geometric shapes and prioritized accordingly in the form of greedy best-first heuristics.** An example of this kind of input is given in the walkthrough examples where the algorithm prefers objects that have proportional width-height values and that are medium sizes. This kind of heuristics can be crucial in handling real-world scenarios where, for instance, dozens of objects might be in a room, ranging from books to kitchenware, but only the large objects such as desks and bookcases can be of use.

- **Objects have symmetries such that the number of contact options in interactions can be decreased by eliminating the redundancies.** In

Chapter 4, as we present how the face-face and face-edge contacts are accounted for in three-dimensional structures, we will point out to the advantage of symmetry and domain rules to eliminate the contact options. Such pruning strategies can help decrease the discrete search space effectively.

- **The maximum number of components that any component interacts with in the structure is minimal** (e.g. 2-3). The connectivity of the interaction graph affects both the efficiency of the constraint satisfaction algorithms and the required input engineering. In terms of the constraint satisfaction, the simplest example is the constraint propagation algorithm which has linear complexity if each variable has only two neighbors [132]. In terms of the input engineering, the stacking domain is a well-detailed example where a stacking action may induce additional constraints beyond the two input object parameters and the domain knowledge must account for such repercussions.

- **Continuous constraints are well-behaved such that linear programming, quadratically-constrained quadratic programming, etc. can be used as a feasibility test.** This property is one of the essential advantages of using a constraint based reasoning as opposed to guided sampling that leads to the commitment problem. The stacking domain and the gear domain presented in Chapter 9 are examples domains with only linear and quadratic constraints respectively and the structure of these constraints can be exploited by solvers such as Simplex and Levenberg-Marquardt optimization.

- **The environment is mostly obstacle free such that validation of output structures and robot symbolic plans through simulation or motion planning has high success likelihood.** The choice of imposing constraints only on the changed pose of the robot as it interacts with the structure (e.g. step sizes) fails to account for the collisions for the environment and different

structures can be sampled by either changing the optimality criteria in Simplex or simply re-running the algorithm with Levenberg-Marquardt, it is ideal to have simpler environments.

3.5.2 Challenging Cases

In this section, we provide a list of scenarios for an input problem that would make it untractable or inconceivable to be tackled by our framework. Note that a number of these items have been expressed in other terms in the list of assumptions and rules in Section 1.1.

- **A component needs to be reused in a different role after the robot starts interacting with the structure.** In assembly planning, a plan where each manipulation action generates a subassembly of the final assembly is called a *monotone plan* [136]. In monotone plans, objects cannot be repositioned in the structure or removed from the structure once they are placed. Such limitations help the planners focus on a smaller discrete search space. The idea of manipulating each object once has also been utilized by Stilman in the Navigation Among Movable Obstacle domain [121]. We enforce a similar rule with the exception that instead of suggesting that an object cannot be moved from a specific position, we insist that the interactions between two objects cannot be nullified once established.

- **Planner has to perform task and motion planning to assemble or use the structure.** The output-based parameterization of motion primitives (e.g. step sizes) is not sufficiently comprehensive to take into account collisions with the environment to perform motion and assembly planning. However, in Section 9.2, we discuss a possible extension to using B-splines as control parameters which, with a small set of parameters, can generate collision-free trajectories

and take into account assembly and use motions in structure design.

- **Planner needs to provide risk guarantees on a structure design before the robot starts using it.** The proposed algorithm does not have a consideration of optimality or risk-aversion but attempts to find any feasible design. In Section 4.5, we deliberate on how to extend our framework with A* and constrained optimization algorithms to take into account uncertainties in object properties and minimize risk.

- **Point contacts are required for structure functionality.** We can claim that point contacts in assemblies in real-world applications are rare since they are unreliable under perturbations or actuation uncertainty. Hence, we make the choice of not including them in the contact search space although the framework can be extended to handle such cases.

- **Planner does not know the physical properties of the objects (e.g. mass, center of mass).** One of the fundamental assumptions of the proposed algorithm is sufficient knowledge about the physics that govern the functionality of the design. However, it is possible to extend our framework into an autonomous system where a robot first experiments with a number of objects in the environments and then having determined their properties plans an appropriate structure.

- **Robot needs to perform a motion that might have dynamic repercussions on the structure stability (e.g. vibrations due to jumping).** Due to the challenges of modeling dynamic interactions between multiple objects in closed form, we choose to focus only on quasi-static robot-structure interactions.

Chapter 4

Static Structures in 2D Translation Domains

4.1 Overview

We begin the application of planning in the constraint space algorithm with 2D translation domains where the components of any structure can only move in xy-coordinates with fixed orientations. The domains with such linear motions mostly include stacking actions such as mixed case palletizing where significant effort has been put forth for optimizing the assembly of rectangular loads in a confined space. However, to the best of our knowledge, the problem has not been considered in terms as a planning challenge where a robot needs to achieve a goal such as climbing a height and needs to construct a structure by using the limited number of box-like objects in its environment.

A secondary motivation for focusing on 2D translation domains is the convexity of the underlying constraints. As we show in Section 4.6, the adoption of the Simplex Algorithm to find feasible samples that satisfy convex constraints leads to very efficient analysis of semantic design proposals. We use this advantage to analyze the

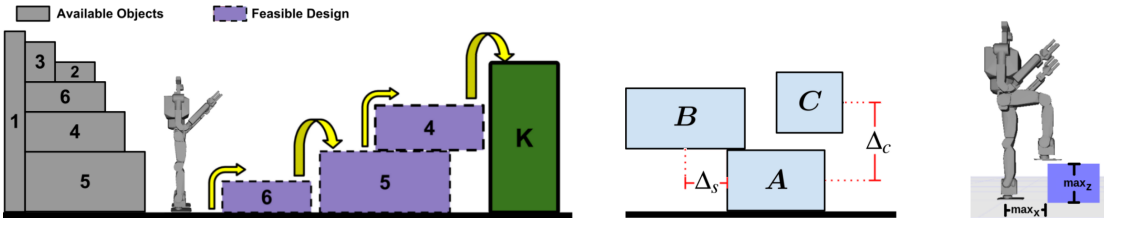
scalability of our approach with complex structures such as overhanging bridges with counterweights.

4.2 Preliminaries

Figure 3a depicts a domain where the goal for Golem Hubo is to climb the obstacle K. Since the object is too tall to step over, one possible option is to construct a stair-like assembly which allows access to it. The assembly needs to fulfill the following constraints:

1. Stability: The stacked objects are stable, even while the robot traverses them.
2. Traversability: The robot path across the assembly is within step size limits.
3. Functionality: The assembly allows the robot to climb on top of object K.
4. Feasibility: The assembly components cannot be in collision.

In the following, we show how the generalized algorithm in Chapter 3 can be utilized to address the aforementioned constraints in the 2D assembly problem.



(a) The robot Golem Hubo is to climb the stair-like assembly

(b) Center of mass and contact stacking constraints

(c) Forward-upward step limits

Figure 3: The three-object convex stacking domain

4.2.1 Structure Stability Conditions

The robot can take two actions in this domain: stacking objects and moving across them. Both actions have implications on the configurations of the two objects *immediately* involved, specifically creating constraints that are functions of the object dimensions and masses, and the robot step sizes and mass. Figures 3b-c demonstrate limitations that are enforced in the constraint functions where (b) stacking implies the two objects are in contact and the COM of the top one is above the bottom and (c) motion implies that the objects involved are within the horizontal and vertical step limits of Golem Hubo. These functions can be written as:

$$\text{contact: } y_C - h_C/2 = y_A + h_A/2 \tag{2}$$

$$\text{stability: } |x_B - x_A| \leq w_A \tag{3}$$

$$\text{horizontal: } (x_A + 0.5w_A) - (x_B - 0.5w_B) \leq max_x \tag{4}$$

$$\text{vertical: } |(y_B + 0.5h_B) - (y_A + 0.5h_A)| \leq max_z \tag{5}$$

where (x, y) coordinates of each object represent its center and COM, and (w, h) represent the corresponding width and height values. The absolute value for the inequality functions imply two separate inequalities. The contact equation states that the height of the bottom of object C should match the height of the top of object A. The stability equation enforces that the object B center lies between the edges of object A. The distances Δ_c and Δ_s in Figure 3b represent the extent of violation of the contact and stability constraints. For the motion, Equation 3 limits the distance between the right edge of the source object and the left edge of the destination object. Similarly, Equation 4 limits the vertical distance between the top faces of the objects.

It is possible to stack multiple objects on top of each other and subsequently, the continuous constraints demonstrated in Equations 1 and 2 need to be generalized beyond the immediate two objects of the actions. Specifically, for an n level

stack of objects, $(n-1)$ stability constraints are needed which enforce that for each object, the COM of the substructure above lies over that object. Every time two objects are stacked, say A and B, the $On(A,B)$ literal is added to the state. Subsequently, a stack of objects starting with object A can be deduced from a chain of $\{On(A,B), On(B,C), ...\}$ literals and done so in line 3 of Algorithm 1. Additionally, the stability needs to be considered as the robot moves across the structure. To ensure the structure does not topple over as the robot moves, for each object stack the robot moves onto, its position at each edge of the top object needs to be incorporated as a stability constraint. Equation 6 represents the constraints that are induced as the robot moves from object A to B, where the sets $Up(O)$ and $Un(O)$ represent the objects in the same stack that are above or under object O respectively:

$$\forall C \in Un(B): \quad \left| \frac{\sum_{D \in Up(C)} x_D m_D + (x_B \pm 0.5 w_B) m_R}{\sum_{D \in Up(C)} m_D + m_R} - w_C \right| \leq 0.5 w_C \qquad (6)$$

where m_R is the robot mass and for each object C under the destination object B, the COM of the subassembly above with the robot is guaranteed to be over object C. Note that the $\pm 0.5 w_B$ term takes into account the robot COM at each edge of the top object and if the robot terms are removed, the stability requirement only for the stacking action is acquired.

4.2.2 Planning Domain Definition

The discrete state is defined with four types of literals: $Used(A)$, $CanGo(A)$, $At(A)$, and $On(A,B)$. $Used(A)$ and $At(A)$ are added respectively if an object A is incorporated to the design and if the robot is on top of that object. $CanGo(A)$ is used as a device to prevent the robot revisiting the same object - once the $At(A)$ is added, $CanGo(A)$ is removed. Table 1 provides the prerequisite and after-effect literals of each domain action, along with the constraints they induce.

The $PutOn(A, B)$ action introduces a new object, A, to the assembly and requires that object B already exists ($Used(G)$ for ground is already in the start state). This action adds the $On(A, B)$ literal that is used with the stability constraints and the equation for the "stable" goal is identical to Equation 5 without the robot term. The $Move(A, B)$ requires that both objects are already used and the robot is at object A. As an after effect, the robot is moved to object B and the literal $At(A)$ is replaced with $At(B)$, and the distance constraints are added between the objects. Finally, to prevent collisions between neighboring stacks, the distance between each pair of objects under the source and destination components is set to a minimum limit of half of the sum of their widths.

4.3 Execution Walkthrough: Pruning Infeasible Configuration Spaces

We study the planning process on the example provided in Figure 3. We begin the forward search with the state $S_0 = \{At(Floor), Used(Floor), \neg Used(o_i) : \forall o_i \in \mathbb{O}\}$ and search for the goal state with the $At(G)$ literal which implies that the robot has reached the top of the obstacle G. $Move(a, b)$ action is prioritized over $Place(b)$ to minimize design complexity and the generated states are analyzed in a decreasing number order for demonstration purposes. The planning graph is denoted with four types of states: final plan states (green), pruned states (orange), backtracked states (red), and unseen states (white). In the following walkthrough, we demonstrate how the planning tree is expanded through the actions defined in Table 1 and how the constraints induced by these actions describe a feasible design for a stair-like structure.

We begin with the first action $Put(6, F)$ where object 6 is placed on the *floor* which adds the $On(6, F)$ and $Used(6)$ literals to the state, and adds three constraints to limit the position of the object on the floor, anywhere except in collision poses

Table 1: The stacking domain action definitions with precondition, after effects and convex constraints

Action	**Preconditions**	**Effects**	**Constraints**	
			Goal	**Equation**
$PutOn(a, b)$	$\neg Used(a), Used(b),$ $\neg At(b)$	$Used(a),$ $On(a, b)$	Contact	$y_a - 0.5h_a = y_b + 0.5h_b$
			Stable	$\forall c \in Un(a)$: $\left\lvert \frac{\sum_{d \in Up(c)} x_d m_d}{\sum_{d \in Up(c)} m_d} - x_c \right\rvert \leq 0.5w_c$
$Move(a, b)$	$Used(a), Used(b), At(a)$	$At(b), \neg At(a)$	Motion	$\lvert(y_b + 0.5h_b) - (y_a + 0.5h_a)\rvert \leq max_z$
			Motion	$((x_b - 0.5w_b) - (x_a + 0.5w_a)) \leq max_x$
			Collide	$\forall c \in Un(a), \forall d \in Un(b)$: $x_c + 0.5w_c \leq x_d - 0.5w_d$
			Stable	$\forall c \in Un(b)$: $\left\lvert \frac{\sum_{d \in Up(c)} x_d m_d + (x_b \pm .5w_b) m_r}{\sum_{d \in Up(c)} m_d + m_r} - x_c \right\rvert \leq 0.5w_c$

with the obstacle. Figure 4 demonstrates the state of the tree, the constraints and the general description of the so-called design at that state.

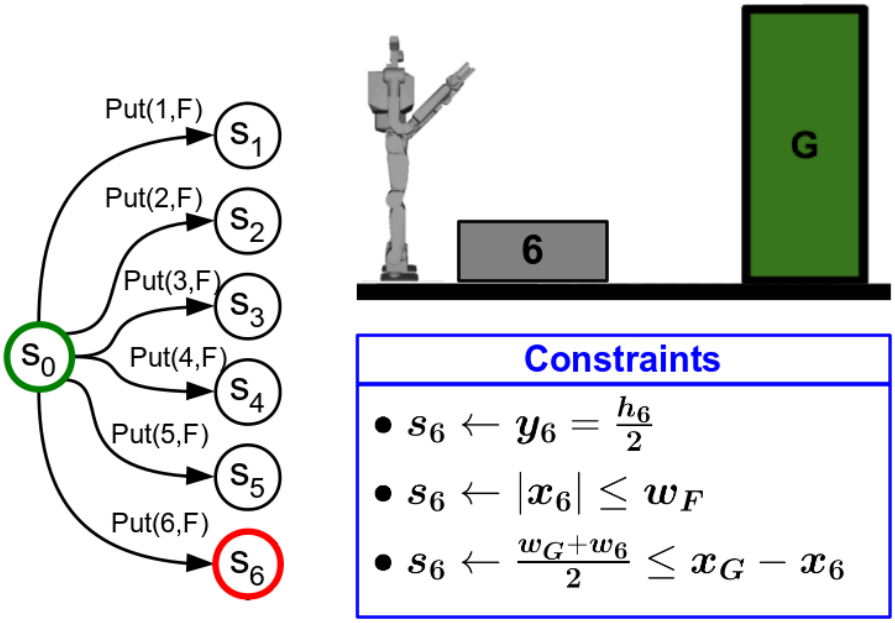


Figure 4: Object 6 is placed on the floor, inducing the first three constraints.

In the next step, the children of the state s_6 are added to front of the queue due to the depth-first search aspect of the planning algorithm and the action $Move(F, 6)$ is chosen due to the aforementioned heuristics. Note that the planner can follow this action, since the newly added constraint $y_6 + \frac{h_6}{2} \leq \max_y$ compares for the step height of the robot to the object height and the feasibility test is returned positive. Observe that this motion feasibility check is only rudimentary in that an exhaustive motion planning algorithm is not executed. Figure 5 demonstrate the updated planning tree along with the state of the design in s_{17}. Another point of interest is that the constraints from the previous action are still maintained in the set of constraints since their validity is crucial to the success of the entire plan.

Once again, the children of s_{17} are added to the queue. However, this time, we observe that the first state s_{23} not feasible because it advocates taking a step from top of object 6 to the obstacle object with the new constraint - an impossible act

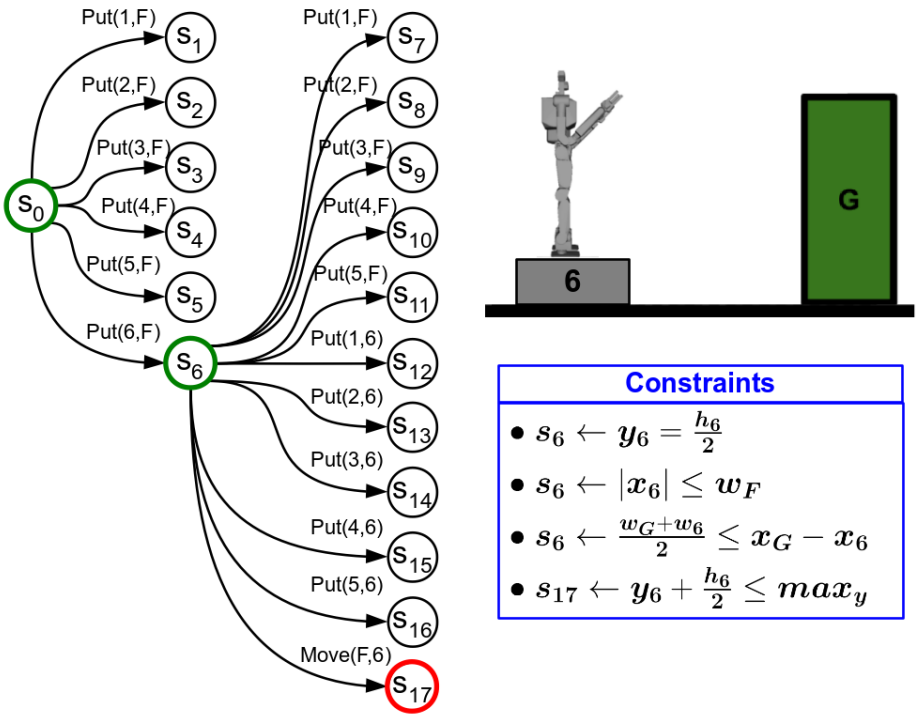


Figure 5: Robot moves on top of object 6 given the step height is within limits.

due to the limited vertical step size. Note that in Figure 6 where we depict the pruned state, we have collapsed the rest of the tree in smaller nodes with indication of the range of states with three dots .... An advantage of embedding continuous constraints in classical planning is the ability to check the feasibility of designs at a detail that is more challenging to capture with abstractions. The feasibility test stops generation of states that do not have continuous assignments that satisfy the accumulated constraints. In this instance, we know any child state of s_{23} would also have unfeasible instantiations and thus, this subtree can be *pruned*.

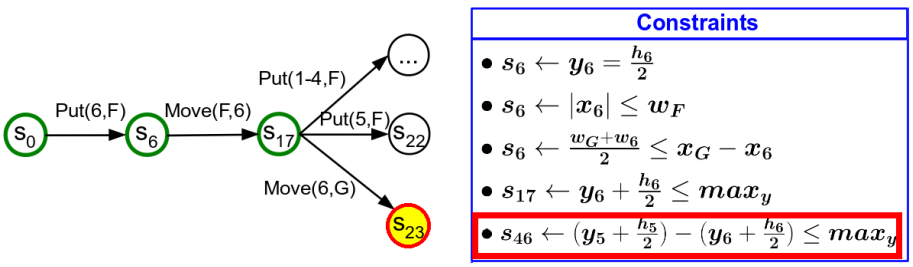


Figure 6: First pruned state s_{23} where action is too large a vertical step.

Moving onto state s_{22} instead, the planner adds the constraints induced by the action $Put(5, F)$, observes that the state is feasible and adds its children to the queue. Note that the constraint $x_6 + \frac{w_6}{2} \leq x_5 + \frac{w_5}{2}$ implies that object 6 is to the left of object 5 and this implementation of this "first object to the left" notion requires the action to inspect the entire state description before introducing the necessary constraints. The constraints and the state in s_{22} can be seen in Figure 7. At this state, the literals are as follows: $\{On(6, F), On(5, F), Used(5), Used(6), At(6)\}$.

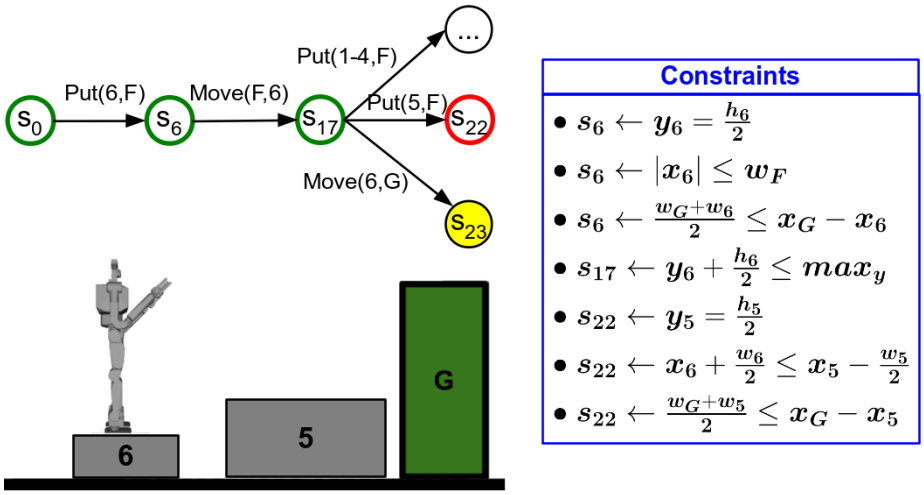


Figure 7: A second object, number 5, is placed on the ground, to the right of object 6 to preserve ordering.

Figure 8 demonstrates multiple events at the same time. First, we observe that the planner chooses state s_{34}, following the action of $Move(6, 5)$. Intuitively, the two possible tactics are either to move from object 5 to the obstacle or create another structure on the floor (cannot be on 5 anymore) and follow that route. However, as the planner finds out, either idea is infeasible and it backtracks, depicting the node with orange in the planning graph. To make a brief walkthrough demonstration, we avoid displaying this exhaustive search. Next option is state s_{33} which is the same as s_{23} and thus pruned. Finally, the action $Put(4, 5)$ is considered (object 6 is not possible because robot is at 6) with three additional constraints. Not that here, the constraint $|x_4 - x_5| \leq \frac{w_5}{2}$ accounts for center of mass of 4 to ensure stability.

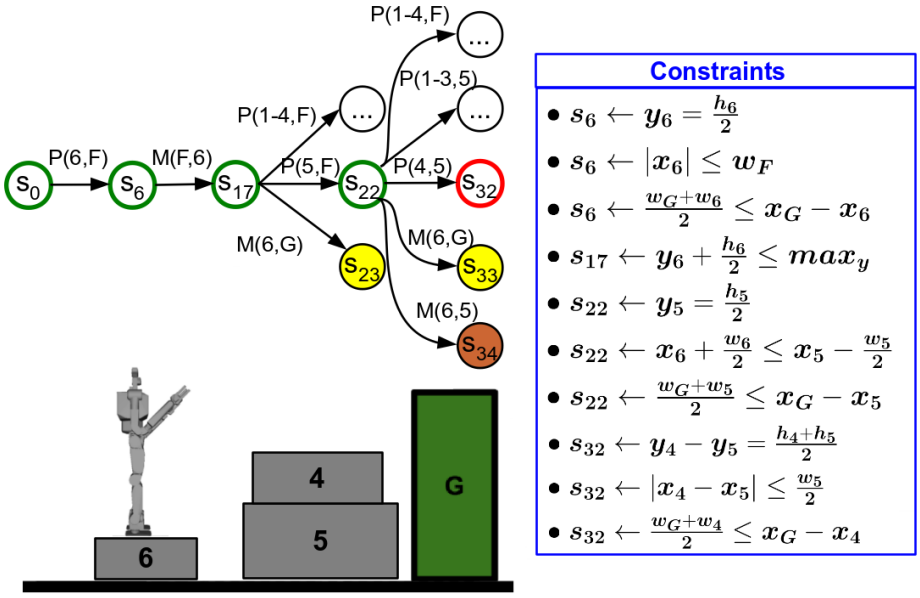
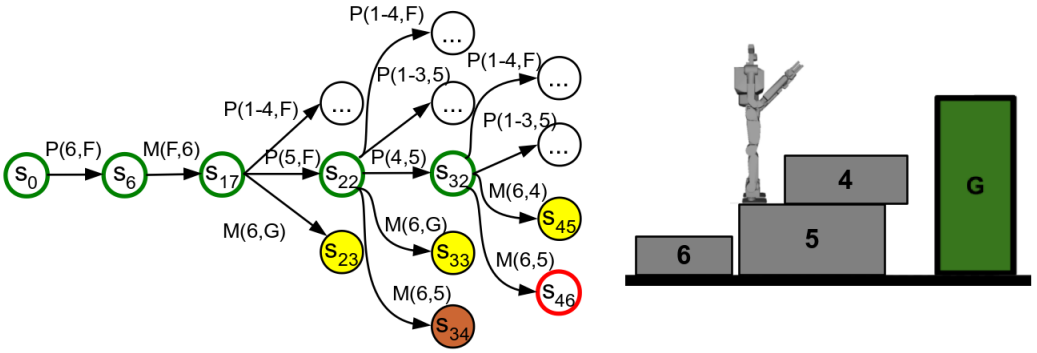


Figure 8: After an exhaustive backtracked search at state s_{34}, the planner decides to stack object 4 on top of object 5.

Next, the planner adds the children of state s_{32} to the front of the queue, where the first action that is preferred is $Move(6, 5)$ similar to the previous step when the action actually had to be backtracked. Figure 9 visualizes the new state along with the three new constraints. Note that the last constraint is particularly of interest due to its complexity in implementation because it ensures that there is enough space on object 5 for the robot to step onto - that is the robot and object 4 would not collide. Moreover, we can observe that the stack composed of objects 4 and 5 automatically has to come closer to object 6 to satisfy the step size constraint. At this stage, the state is composed of: $\{On(6, F), On(5, F), On(4, 5), Used(4), Used(5), Used(6), At(5)\}$.

Moving forward, only two more actions, $Move(5, 4)$ and $Move(4, G)$, are left until the $At(G)$ literal is added to the state and the planning can be stopped. To side with brevity, we now only present the final planning graph in Figure 10 and the final state of the design. Observe that the placement of object 4 also takes into account the mass of the robot as detailed in Table 1 and that is the reason, its center of mass is sufficiently inside the support of object 5.



Constraints

- $s_6 \leftarrow y_6 = \frac{h_6}{2}$
- $s_6 \leftarrow |x_6| \leq w_F$
- $s_6 \leftarrow \frac{w_G+w_6}{2} \leq x_G - x_6$
- $s_{17} \leftarrow y_6 + \frac{h_6}{2} \leq max_y$
- $s_{22} \leftarrow y_5 = \frac{h_5}{2}$
- $s_{22} \leftarrow x_6 + \frac{w_6}{2} \leq x_5 - \frac{w_5}{2}$
- $s_{22} \leftarrow \frac{w_G+w_5}{2} \leq x_G - x_5$
- $s_{32} \leftarrow y_4 - y_5 = \frac{h_4+h_5}{2}$
- $s_{32} \leftarrow |x_4 - x_5| \leq \frac{w_5}{2}$
- $s_{32} \leftarrow \frac{w_G+w_4}{2} \leq x_G - x_4$
- $s_{46} \leftarrow (y_5 + \frac{h_5}{2}) - (y_6 + \frac{h_6}{2}) \leq max_y$
- $s_{46} \leftarrow (x_5 - \frac{w_5}{2}) - (x_6 + \frac{w_6}{2}) \leq max_x$
- $s_{46} \leftarrow (x_4 - \frac{w_4}{2}) - (x_5 - \frac{w_5}{2}) \leq foot$

Figure 9: The robot moves onto object 5. The constraints capture the fact that object 4 has to be shifted to make room for the robot and yet stay within the support.

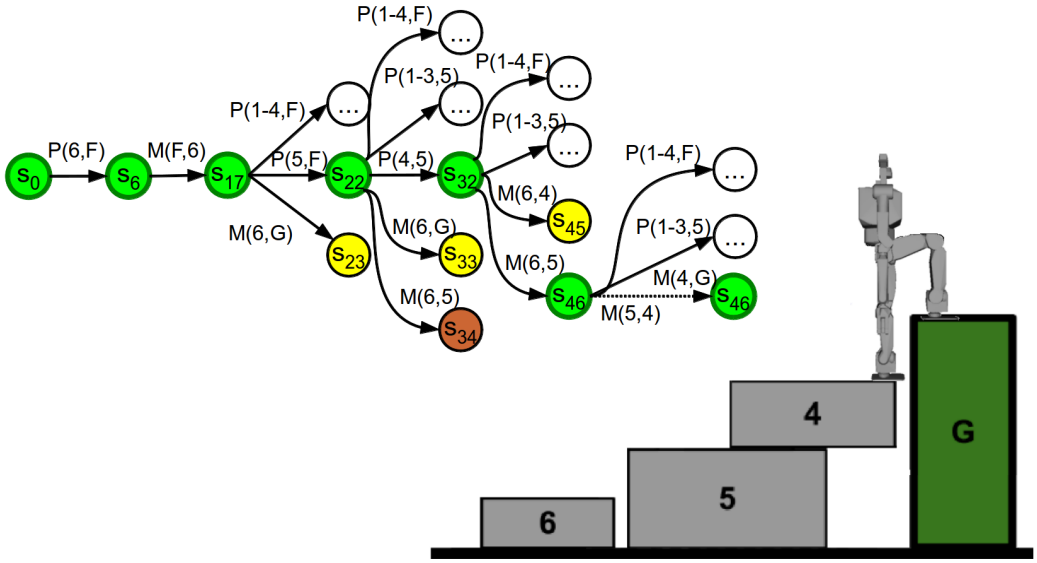


Figure 10: The final planning graph, along with the state of the successful design.

So far, we have discussed a case where the heuristics are hand-tailored to demonstrate the simplest walkthrough (despite the backtracking instance) and a solution exists. As the planner searches pops states of the queue, if a state with a $At(G)$ literal

does not come up and the queue is emptied, then the planner would conclude that a design is not feasible with the given objects for the input task.

4.3.1 Visualizing the feasible space

Figure 11 demonstrates the space of x-axis coordinates of the assembly components as the plan progresses forward. With the placement of the first object, which enforces that it is positioned to the left of the others, the constraints $x_6 < x_4$ and $x_6 < x_5$ are imposed. Secondly, when object 4 is placed on object 5, its center of mass is limited to be on object 5 and thus, the two values x_4 and x_5 are coupled. While the third action, the robot moving from object 6 to 4, leads to a minor difference, the greatest change is seen with the last action as the robot moves from object 4 to the goal. To enable the move, object 4 is pulled towards the goal (larger x-axis values) and the effect is seen clearly with the cut off in the vertical axis. The cut off is parallel by the shift in the +x direction in the last two assembly states, S_{46} and S_{53}.

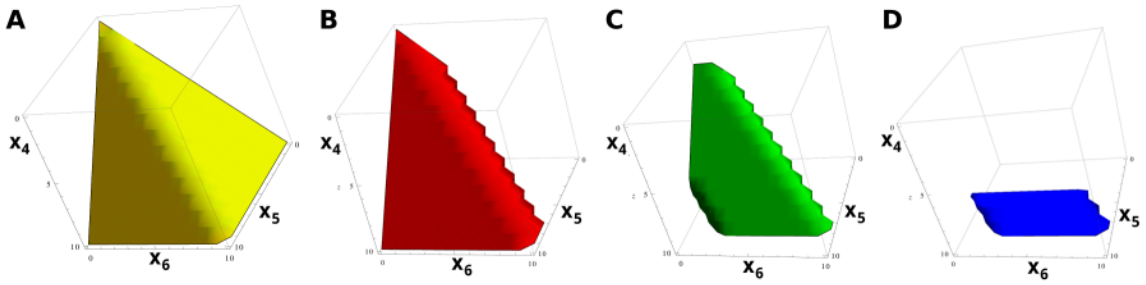


Figure 11: The solution space for the x-axis coordinates shrinking as the plan progresses and constraints accumulate

This visualization of the feasible subspaces and understanding the effects of the constraints on the volumes of the subspaces will prove to be crucial in generating domain-independent heuristics for the constraint-based planners. The key idea that is presented in Chapter 8 is that if we can determine which actions are more effective in shrinking the feasible space, we can prioritize them to either find the goal states or prune the planning tree effectively.

4.4 Golem Hubo Simulation Constructing a Bridge

A drawback of the proposed approach is that the functionality of the output assemblies are strongly dependent on the expert user's modeling of the problem. For instance, if the user does not incorporate the robot mass in a bridge design composed of stacked objects, the structure might collapse during execution time. Moreover, there is an open question as to what level of detail is sufficient in modeling the environment. To evaluate the fidelity of the constraints in representing the stability rules and the robot constraints, we have run experimental results with Golem Hubo in *dynamic simulation*. The example problem shown in Figure 12 is chosen to push the limits of the algorithm both in the number of available objects and their configuration in the sense there is a wide inaccessible region on the floor (e.g. fire) and a tall obstacle to climb.

Figure 12 demonstrates the design of a bridge-like structure and its traversal by Golem Hubo in dynamic simulation. The planner uses the previously defined domain actions with the addition of a constraint that prevents objects from being placed in the inaccessible region. The DART library is used to simulate the multi-body kinematics using the Lagrange's equations derived from D'Alemberts principle [85]. An interesting question that arises as we attempt such applications is the feasibility of such designs in terms of their autonomous construction.

4.4.1 Buildability

Reasoning about collisions in an assembly problem is crucial for the soundness of any proposed planner. Previous work has utilized heuristic distance functions that bound the distance between possibly colliding objects such as the fulcrum and the robot. The tradeoff is that such functions overconstrain the configuration spaces. We have also considered using geometrical plane fitting between convex objects as a method

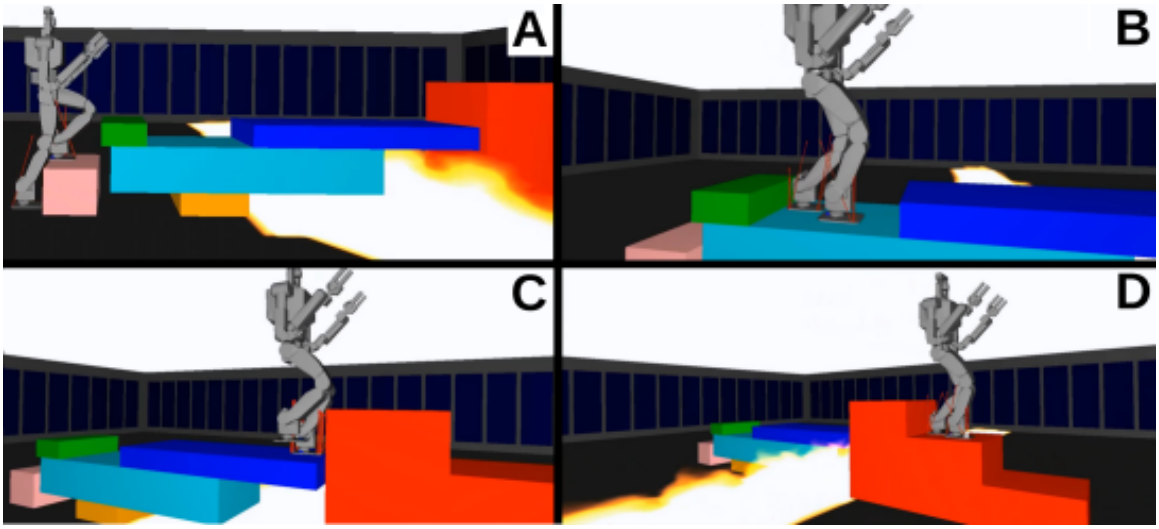


Figure 12: Hubo crossing fire with the designed bridge in dynamic simulation. Arrows at the feet represent the contact forces.

to ensure they are collision free [63], however, the number of extra variables for the plane parameters slow down the overall approach. To the best of our knowledge, incorporating collision avoidance in a constraint framework is not trivial. Moreover, an approach to solve the harder problem of guaranteeing that the construction is collision-free may be simpler and more useful, although less efficient. In fact, we argue that *the design of a simple machine must consider the robot motion during its construction; otherwise, its feasibility cannot be determined.*

4.5 Incorporating Risk in Design Choices

An important aspect of reasoning about creating structures that can extend a robot's configuration space is the safety of the robot as it interacts with the structure. The aforementioned example where the robot uses an overhanging bridge to cross a fire is a simple example where in a failure case, the safety of the robot may be in peril. In this section, we briefly discuss possible extensions to our framework that take into account the possibility of failure in design choices.

We claim that the key risk factors behind failure cases are **perception and actuation uncertainties** where the robot may either not know the exact dynamic/geometric properties of the objects or may not execute a desired behavior correctly. In the field of mobile manipulation, Thrun et al. [125] has attempted to address this issue by augmenting their particle filters to track more relevant hypotheses with a risk-based cost model. Similarly, in motion planning for partially-observable domains, [5] and [102] have suggested using so-called **risk-sensitive POMDP models** with appropriate **utility functions** to guide the action-level search. Note that these two studies have tackled the challenged of incorporating high-dimensional nonlinear utility functions differently by either sampling directly the continuous MDP models with Monte Carlo Value Iteration [5] or using a linearized version of the utilities [102]. Another common approach is to augment the classical **A* search** algorithm with a necessary domain-dependent cost function that computes the probability of failure [57, 110]. Finally, the idea of **one-switch utility** functions is introduced by Liu et al. in [91, 92] where an agent tends to become risk-neutral as its "wealth" increases in a decision-theoretic sense despite starting off as risk-averse. This could be relevant to structure design in search and rescue operations when an agent has limited time or objects to generate a structure and has to balance its resources against its evaluation of risk.

An analysis of the literature suggests that the majority of the approaches [5, 102, 110] focuses on computing an optimal plan which minimizes a risk utility function. This is in contrast to our framework where the goal is to find any feasible structure that satisfies the goal constraints while respecting the kinodynamic limitations of the robot. In order to generate an optimal solution for the design problem, we can adopt a two-layered solution where in the symbolic planner, an A* search algorithm prioritizes the search with a certain heuristic evaluation of the design proposals. The heuristic value can be obtained by solving a **constrained minimization** problem as opposed to the general constraint satisfaction problems we have proposed so far.

The inclusion of the minimization is similar to the use of the utility functions in the literature where the cost express the uncertainty in the design operation. The simplest problem case for such an approach would be when the uncertainty distribution over the object properties such as mass and center of mass are known to the planner and thus, a utility function in terms of the uncertainty parameters and the design variables can be formed. Note that the *commitment problem* can still be avoided by iteratively resolving constrained minimization problems without committing to specific minimizer poses in the previous state.

Another scenario of interest might be when an uncertainty model is not available to the planner but a set of examples of successful and unsuccessful designs are present. The concept of learning from given examples is an intriguing one for the structure design domain because an intuitive connection to early childhood learning with block examples arise. A number of studies have been conducted on children's abilities to replicate Lego block models [2, 12]. A key idea that might lead to an extension of our work is that if a design can be decomposed into its contact points such that a constraint graph can be generated, then a similarity metric between two designs can be based on the similarity of the graphs and the similarity of shapes. A number of **shape similarity** measures are available, including Minkowski distances, [130] and **graph topology similarity** is a well-studied subject [50, 114, 144]. Using these two metrics, a heuristic function can be proposed that prioritizes choices that lead to designs similar to previously observed positive outcomes, rather than negative ones.

4.6 Conclusion

One advantage of the proposed framework over methods that discretize the configuration space and commit to particular samples is its ability to reason about entire configuration subspaces. This property is particularly highlighted in convex domains

where the simplex algorithm is very efficient. For instance, the feasibility test for a scenario where ten objects are used to construct a stair-like structure using 10 equality and 46 inequality constraints takes only 2 milliseconds (10 trials) regardless of whether the constraints can be satisfied with the given object dimensions and obstacle height.

Although we have exploited the convexity of the 2D linear translation domains, most functional structures live in 3D spaces with components that have full six degree of freedom configurations. In the next chapter, we start our analysis of static structures with nonlinear orientation and distance constraints.

Chapter 5

Static Structures with 6-DOF Components

5.1 Generalization to 3D

The proposed algorithm presented in Section 3.3 is indifferent to the input planning domain and the accompanying constraints. It is the user's responsability to encode the semantic domain relationships as physical constraints that can be enforced in the configuration space of the structure components. In this section, we discuss how the structure stability conditions presented in Section 4 in 2D translation domains can be generalized to 3D.

As the objects can be positioned in any 3D configuration, the problem definition has to address their orientations. First of, the object orientations naturally raises the problem to a nonlinear domain since the object face and edges that make contact are now trigonometric functions of their respective poses.

Secondly, the number of possible contact types between objects increases when compared to the only face-to-face type that was accounted for in the 2D translation

domain. In general, two polyhedral objects can make contact on the following locations: (1) vertex-to-vertex, (2) vertex-to-edge, (3) edge-to-edge, (4) edge-to-face and (5) face-to-face. The first three of these types are only point contacts that we prefer not to include in domain definitions since the stability of point contacts are vulnerable to small perturbations. Instead, in this work we only focus on line and plane contacts. In the following section, we define the equations for line contacts.

5.2 Equations for Line Contacts

We now present contact models for objects which may be posed in any 3D configuration and have a multitude of faces/edges to interface with. We model only face-to-edge contacts since only edge-to-edge or edge-to-point contacts are not as robust to disturbances. Figure 13 demonstrates the parameters involved when a face (red) of lever L (blue) makes contact with an edge (black) of fulcrum F. Let ${}_wP_F^1$ and ${}_wP_F^2$ represent the positions of the two fulcrum edge and let ${}_wP_L^1$ to ${}_wP_L^4$ represent the positions of the lever face vertices.

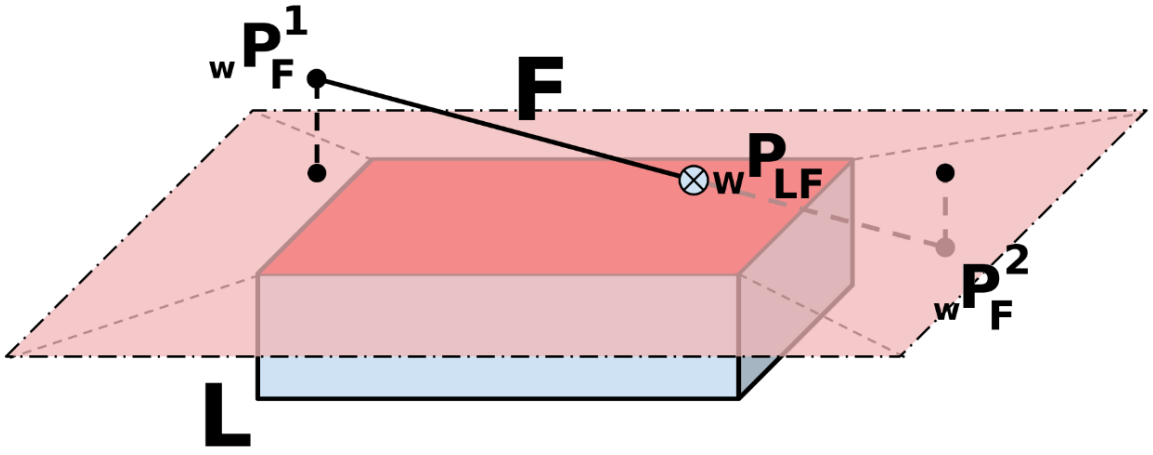


Figure 13: The variables involved in the contact constraint between a face of a lever L and an edge of a fulcrum F.

The concept of contact encompasses two notions. First, the distance between the edge endpoints to the face plane (pink) should be zero. Second, the edge and the face

should share a point. Otherwise, even though the edge might lie on the face plane, it would not necessarily be in contact. Plane parameters with normal w^n and distance d from origin are computed from the face vertices:

$$\begin{aligned} {}_w n &= ({}_w P_L^3 - {}_w P_L^1) \times ({}_w P_L^2 - {}_w P_L^1) \\ d &= {}_w n \cdot {}_w P_L^1 \end{aligned} \tag{7}$$

Then, the distance from the fulcrum edge endpoints to the face plane should be zero:

$${}_w P_F^1 \times [{}_w n, d] = {}_w P_F^2 \times [{}_w n, d] = 0 \tag{8}$$

Secondly, the contact point ${}_w P_{LF}$ has to lie both on the face and the edge. For the edge, the point has to lie on the line between ${}_w P_F^1$ and ${}_w P_F^2$ (Equation 9), and be within the line segment (Equations 10-11). Let $\vec{f}$ represent the normalized vector from ${}_w P_F^1$ to ${}_w P_F^2$.

$$||((\vec{f} \cdot ({}_w P_{LF} - {}_w P_F^1)) * \vec{f}) + {}_w P_F^1) - {}_w P_{LF})|| = 0 \tag{9}$$

$$(\vec{f} \cdot ({}_w P_{LF} - {}_w P_F^1)) \geq 0 \tag{10}$$

$$(\vec{f} \cdot ({}_w P_{LF} - {}_w P_F^1)) \leq ||{}_w P_F^2 - {}_w P_F^1|| \tag{11}$$

Finally, the contact point should lie within the face. Let ${}_w n_L^i$ be the unit normal that faces inside the mesh for each edge i, $1 \leq i \leq 4$. The vector from each point to the contact point should have a positive projection along the matching normal:

$$({}_w n_L^i \cdot ({}_w P_{LF} - {}_w P_L^i)) \geq 0 \quad \text{for } \ 1 \leq i \leq 4 \tag{12}$$

Note that the positions expressed in the world coordinates in this presentation need to be derived from the object poses and vertex locations in the local object frames.

Lastly, the presented face-to-edge constraints are parameterized by the face and the edge choices. Although this parameterization allows relatively simple expressions of complex contact constraints, search in the space of faces and edges of available

objects can be untractable. To improve scalability, we can exploit shape symmetries (see Section 6.3). For instance, if two faces of an object are known to serve the same purpose around an axis of symmetry, one can be omitted from the set of possible parameters.

5.3 A Minimization Approach for Nonlinear Constraint Satisfaction

In the previous sections, we have introduced nonlinear constraints for robot kinodynamic limitations and 3D object contacts. Any of these constraints can be interpreted as an error function which returns a nonzero value if violated and zero otherwise. For the feasibility test in nonlinear domains, we propose expressing the constraints as error functions where we use the square of the induced errors due to a pose $\vec{x}$:

$$f(\vec{x}) = 0 \;\Rightarrow\; E_f(\vec{x}) = f^2(\vec{x})$$

$$g(\vec{x}) \leq 0 \;\Rightarrow\; E_g(\vec{x}) = \begin{cases} g^2(\vec{x}) & \text{if } g(\vec{x}) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Next, given a set of equality and inequality constraints $\mathcal{F}$ and $\mathcal{G}$, the total error is:

$$\mathcal{E}(\vec{x}) = \sum_{f \in \mathcal{F}} E_f(\vec{x}) + \sum_{g \in \mathcal{G}} E_g(\vec{x}). \tag{13}$$

If the total error $\mathcal{E}(\vec{x})$ is zero for some configuration assignment $\vec{x}$, there exists a design that satisfies all the constraints. The feasibility test attempts to find the global minimum of the total error function and checks whether it is zero. If a zero minimum is not found, the feasibility test returns false and the algorithm backtracks. We use the Levenberg-Marquardt method [107] to find local minima and for the global minimum, the algorithm is initialized with a number of random seeds. The random

restarts from a bounding box approach leads to a probabilisticly complete feasibility test. We used the efficient Levenberg-Marquardt implementation in GTSAM [24].

5.3.1 Backtracking Feasibility Test for Informed Initializations

Sampling random seeds for local optimization becomes a crucial drawback of the minimization approach as the number of variables increases. Within a planning context, the number of variables and the constraints increase as the plan grows. *A useful observation is that any feasible configuration in the goal state would also satisfy the partial assembly constraints in the earlier states.* Secondly, the earlier partial assemblies are easier to compute since the number of constraints is smaller.

We propose to instantiate the variables by incrementally solving for the constraints in the parent states and using their results as seeds in minimization. If a variable is introduced in a state (e.g. it does not have a solution in the parent state), it is sampled from a predefined bounding box provided by the user. Let $\mathbb{C} = \{C_1 ... C_k\}$ be the sequence of k sets of constraints where C_i is the constraint set of the i^{th} state in the plan. The goal of the feasibility test is to determine if the set C_k is satisfiable. Algorithm 2 begins by attempting to optimize the input state's constraints by initializing its variables either with the provided assignments from parent state or with random samples (lines 4-9). The minimize function in line 8 implements Levenberg-Marquardt with the cost function in Equation 13 and the initial values X_i. If a solution exists, the algorithm provides it to the successive state. If the successive state returns with an assignment (line 14) or the current state is the last one in the plan(line 11), return the assignment. Otherwise, repeat the optimization with new random samples (line 13) or backtrack into previous state (line 15).

Algorithm 2: NonlinearFeasibilityTest($i, \mathbb{C}, X_{i-1}, MAX_{seed}, MAX_{fail}$)

Input: i: index into plan, $\mathbb{C}$: sequence of plan constraints, X_{i-1}: solution to parent state constraints, MAX_{seed}: maximum number of seeds, MAX_{fail}: maxium number of failures before backtracking

Result: X: functional assembly configuration

```
1  v_i ←var(ℂ[i]), v_{i-1} ←var(ℂ[i − 1]);        // Current and parent state variables
2  for failures ← 0 to MAX_fail do
       // Attempt to optimize the current state's constraint cost function
3      X_i ← ∅;
4      for random ← 0 to MAX_seed do
5          foreach var ∈ v_i do          // Instantiate with parent result or sample
6              if var ∈ v_{i-1} then X_i[var] ← X_{i-1}[var];
7              else X_i[var] ←randomSample();
8          [error, X_i] ←minimize(ℂ[i], X_i);
9          if |error| < ϵ then break;  // If error is 0, all constraints are satisfied.
10     if X_i = ∅ then return ∅;                    // If state is infeasible, return.
11     if i = |ℂ| then return X_i;      // Return succcesful result in the last state.
       // Minimize successive state constraints with current assignments as partial
          initialization.
12     X_{i+1} ←NonlinearFeasibilityTest(i + 1, ℂ, X_i, MAX_seed, MAX_fail);
13     if X_{i+1} = ∅ then continue;      // If failed, provide a new seed to the child
       state.
14     else return X_{i+1};                                    // Success case.
15 return ∅;          // If failed to feed useful seeds to the successive state, return.
```

5.4 Constructing Ramps

Having established a method to evaluate nonlinear constraints, we now turn to an example where Golem Hubo needs to climb a height using wooden boards and cinder blocks. The main difference in this example is that (1) objects have full three-dimensional orientations and (2) the configuration of the objects are defined by the face or edges that they use to make contact with other objects. For instance, a cinder block can be placed on the ground on all of its faces except the ones with the holes, which limit all the yaw and pitch values around the ground frame but it can still rotate freely around the vertical axis. Similarly, when a board is placed on the ground,

it can only rotate on the ground plane but can not take any actions that would break the contact with the ground.

Next, we define the domain with the state literals, abstract actions, start and goal states, and the continuous constraints. First, the domain has the literals: $Used(a, b)$, $IsBoard(a)$, $IsCinder(a)$, $IsObstacle(a)$, $IsGround(a)$, $Connected(a, b)$. The goal state has the literal $Connected(ground, obstacle)$ and the initial state has definitions for the objects in the scene describing their types: board, cinder, obstacle or ground. The goal of the planner is to apply the domain actions to connect the objects in the environment so that there is a path from the ground to the obstacle that the robot can traverse. The available actions are $SupportBoard(a, b)$, $PlaceBoard(a, b)$ and $PlaceCinder(a, b)$. Figure 14 displays the effects of these actions for some parameters.

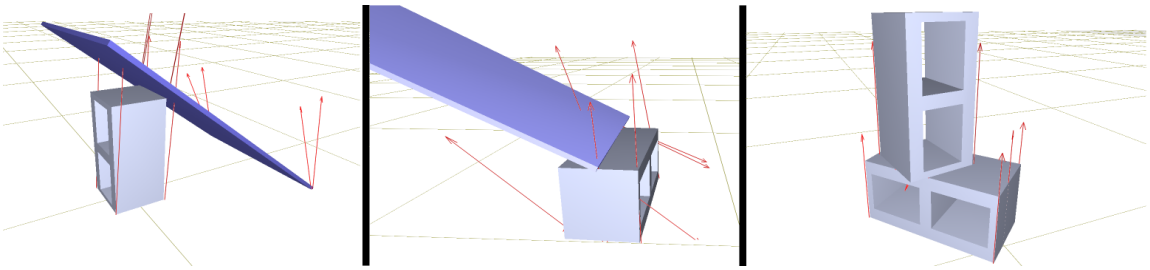


Figure 14: Domain actions: supporting the board vertically, placing the board on a surface, and stacking cinder blocks

Lastly, we have to provide the domain knowledge to the planner so that it can reason about the effects of the abstract actions on the continuous configuration space of the design components. We define the following three constraints: edge-face contact (i.e. board edge lying on ground), face-face contact (i.e. cinder block on ground), and center of mass-face projection (i.e. cinder block resting on cinder block).

5.4.1 Ramp Domain Constraints

We first define some terminology that will help the following formulations. For an object o_a, the homogeneous transformation T_a^0 defines the pose of the object in the

world frame (denoted with 0). Let E_i^a denote the i^{th} edge of the object mesh M_a and let F_j^a denote the j^{th} face. Then, the variables $E_i^a(0)$ and $E_i^a(1)$ represent the two edge points defined in the *local* frame of the object - that is, the world coordinates of the first point would be $T_a^0 E_i^a(0)$. These definitions are important because most of the time, the configurations of the objects need to be related to one another through local edge and face information, and to do so, the local coordinate frames need to be represented in the *common* world frame. Similarly, the position of the first vertex of the face j in the world frame is represented by $T_a^0 F_j^a(0)$.

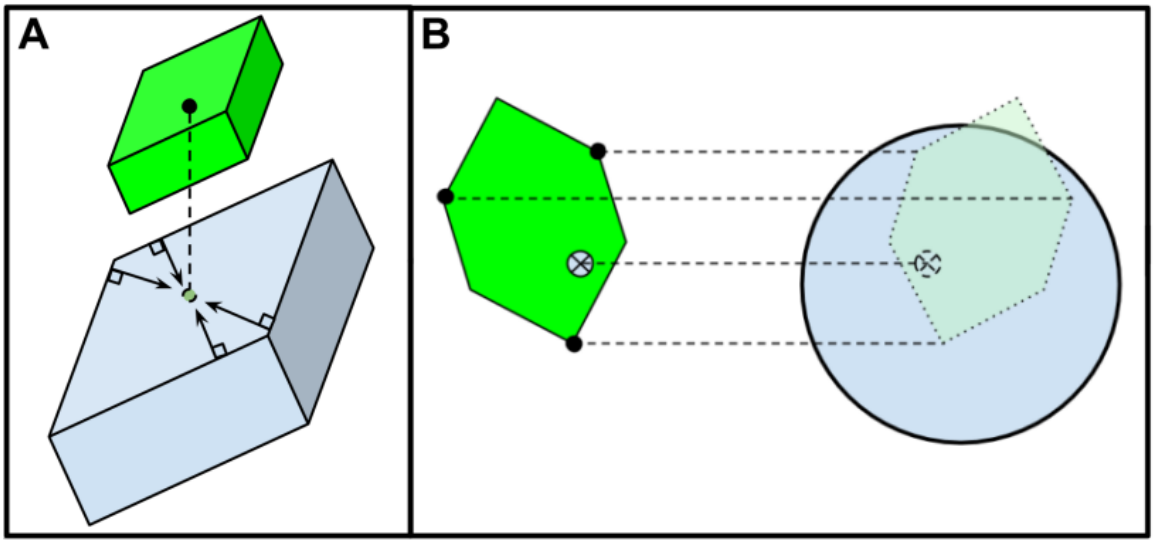


Figure 15: COM constraint and face-to-face constraints for stacking actions

The constraints are described as follows:

1. **Center of mass-face projection**: $c_{cf}(o_a, o_b, i)$: This constraint ensures that the projection of the center of mass of object o_a lies within the face F_j^b of object o_b. Geometrically, the constraint requires that the projection of the center of mass on the face plane, say P, has a positive projection to the normals of edge of the face. Otherwise, the projection would be outside the face, having crossed that particular edge. Figure 15a demonstrates the edge normals and the projection of the center of mass of the top object being placed on the bottom.

Thus, the constraint returns the distance between the projected center of mass and the edge that it crosses. If for a given edge, the center of mass is within the polygon, then the constraint incurs zero error.

2. **Edge-face constraint**, $c_{ef}(o_a, o_b, i, j)$: In Section 5.2, we introduced the equations for the line contact as an example of generalization of contact constraints to 3D. The key idea is that a set of simple projections with equality and inequality functions can be utilized to ensure that the edge lives on the plane of the face and the two objects share at least a single point in common.

3. **Face-face constraint**, $c_{ff}(o_a, o_b, i, j)$: Given the faces of the two objects o_a and o_b, this constraint enforces that the faces are in contact, meaning that they share the same plane and a common contact point. The expression of the error function is very similar to the edge-face constraint where (1) error values for the contact point sitting within the surfaces is returned and (2) instead of the edge endpoints, the distances of three vertices of one of the faces to the face plane of the other is used as error criteria.

5.4.2 Planning Domain Definitions

The three constraints defined above are essential to understanding the underlying formulations that govern the ramp examples and the lever-fulcrum examples in the next section. The geometric projections presented ensure that the objects are configured in poses where they can support each other's weight and transmit force if necessary (e.g. lever). Now, we formally define the actions in the ramp domain in Table 2. The most significant aspect of the table is the connections added at each action which eventually create a pathway from the ground to the obstacle.

Table 2: The ramp domain action definitions with precondition, after effects and constraints

Action	**Preconditions**	**Effects**	**Constraints**
$SupportBoard(a, b, j)$	$IsBoard(a), Used(a), Used(b)$ $IsCinder(b) \vee IsObstacle(b)$	$Connected(a, b)$ $\forall c$ s.t. $Connected(b, c) \rightarrow Connected(a, c)$	Edge-Face(a,b,0,j)
$PlaceBoard(a, b, j)$	$IsBoard(a), \neg Used(a), Used(b)$ $IsCinder(b) \vee IsGround(b)$	$Used(a), Connected(a, b)$ $\forall c$ s.t. $Connected(b, c) \rightarrow Connected(a, c)$	Edge-Face(a,b,2,j)
$PlaceCinder(a, b, i, j)$	$IsCinder(a), \neg Used(a), Used(b)$ $IsCinder(b) \vee IsGround(b)$	$Used(a)$	Face-Face(a,b,i,j) COM-Face(a,b,i,j)

5.5 Results

In addition to the geometric constraints that govern the rules for assembly, we formulate the ramp problem with constraints due to the robot kinematics and the object properties. First, the humanoid robot Golem Hubo is assumed to have a maximum climbing angle of 35 degrees which limits the orientation of the boards. Secondly, the problem is set up with two cinder blocks and two boards, with the goal object height at 1.2 meters. The two boards are 1.4 and 1.0 m long which suggests that only the first board can be used to reach the obstacle, if and only if the cinder blocks are used to support it.

The left section of Figure 16 demonstrates the constraints accumulated throughout the output plan with each action step denoted next to the constraint. The plan first stacks the cinder blocks, then places and supports the small board, and finally completes the structure with the larger board that connects the cinder blocks to the obstacle. On the right, we display the keypoints from the traversal of the structure autonomously in dynamic simulation by Golem Hubo.

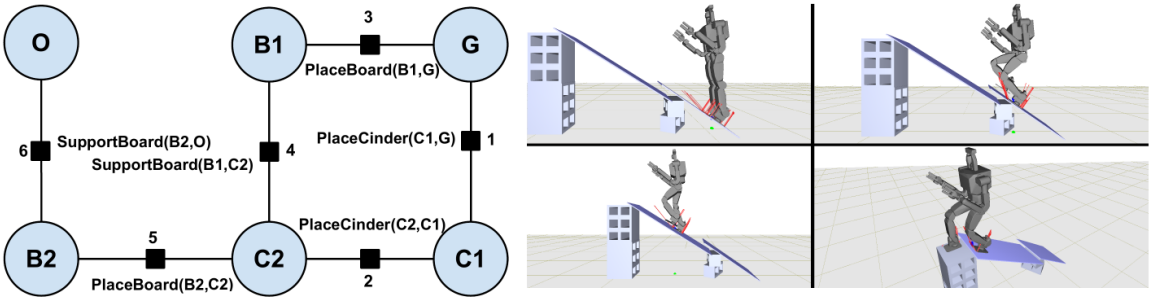


Figure 16: The constraint graph for the ramp structure

A crucial concern in the step towards fully nonlinear steps that also incorporate the robot kinodynamics is the efficiency issues due to the random restart approach with the iterative local optimization method, Levenberg-Marquardt algorithm. In this problem, with less than 20 error values accumulated between 6 constraints, the

average runtime for the feasibility test is 24 milliseconds, tested over 20 trials. Although the timing is at least an order of magnitude slower, an interesting point is that the number of restarts for each trial is at most 2, suggesting that the efficiency breakdown is not due to the locality of the optimization approach but simply the complexity of the nonlinear problem.

5.6 Real Life Experiments

To demonstrate how well the constraints imposed by the planning framework captures the requirements of creating a complex ramp structure, we ran experiments using Golem Krang and a toy car as shown in Figure 17. Similar to the simulation results in the previous sections, the goal for the planner is to create a structure that the RC car can use to drive up to a height. Just like Golem Hubo, the RC car can climb a limited angle of incline which constraints the ramp poses.

To create the structure, Golem Krang was teleoperated to move cinder blocks and wooden plates into the configurations output by the planner. The experiments demonstrate that in addition to the feasibility of proposed designs, the robot has the manipulation capabilities to construct the structure if an appropriate motion plan can be computed. We have made progress in the autonomous construction of such a system and plan to conclude it in near future.

Figure 17: Real life construction of a ramp-cinder block domain

5.7 Collision Avoidance with Nonconvex Shapes

Although the collision avoidance in 2D stacking domain has been expressed clearly in terms of object heights and their x-axis positions, in 3D, with an increased search space of face and edge contacts, we have so far glossed over the issue in this presentation. One of the fundamental assumptions of our work as expressed in Chapter 3, Section 1.1.2, is that the objects are convex. Note that, to ensure the cinder blocks in this examples abide by this restriction, their bounding boxes were used as geometric inputs in the domain definition in Table 2. Similarly, to work around the complex collision detection between the robot and the fulcrum in Chapter 6, a distance function is imposed so that the two objects are guaranteed not to collide as long as they are sufficiently distant from each other.

The advantage of the convexity assumption is that two convex objects can be ensured to not collide as long as there exists a separating plane. In other words, the problem of collision detection can be expressed as a plane fitting problem. In our preliminary work, we have experimented with incorporating plane parameters in the constraint satisfaction problems with positive feasibility results. In fact, the idea of incorporating contact and collision constraints within optimization/feasibility frameworks is a recently popular approach in computer graphics and motion generation [106].

In generalizing our work to nonconvex shape inputs, the primary approach would be to adopt well studied convex decomposition methods [80, 90]. The key idea is that each nonconvex object is decomposed into multiple convex parts and the separability of the two nonconvex objects is ensured by generating a combinatorial number of separating planes between each pair of convex components. Although theoretically this is a viable solution, in practice, we foresee challenges in computational overhead since the complexity of the shapes could lead to a high number of plane parameters and separation constraints, decreasing the efficiency of the feasibility tests.

5.8 Conclusion

We demonstrated how the planning framework demonstrated in Chapter 3 and applied to static structures in 2D linear domains can be generalized to components with six degree of freedoms. The main focus was on how the orientations of the objects induce nonlinear constraints. We show that the minimization of the errors due to the violation of constraints can be used as a constraint feasibility test instead of the Simplex algorithm for convex domains adopted previously.

Chapter 6

Quasi-Static Structures: Simple Machines

In Chapter 4 and 5, we proposed using different constraint satisfaction algorithms within the same search-based planning algorithm to find feasible configurations for static structures. All of the structures such as bridges and ramps were created to help the robot reach a location, i.e. climb a height, where the simplest robot model was utilized either using step size or maximum incline limits. In this chapter, we show that in addition to static structures, the proposed planner can reason about simple machines that have moving parts. Note that the motion of the structure, for instance that of a lever-fulcrum assembly, can be analyzed only in terms of the force and torque equilibriums - hence a quasi-static analysis is sufficient. Moreover, to enable more complex manipulation tasks, the robot limitations need to be considered in depth, taking into account joint torque limits, reachability and posture control.

6.1 Kinodynamic Constraints

In designing a structure for a robot to manipulate, the planner needs to reason about the physical capabilities of the robot. Figure 18 visualizes two types of constraints

for the kinematics and the posture of Golem Krang. First, the contact point on the lever needs to be in the reachable space of the robot. Note that in addition to the position requirement, the orientation of the gripper should also be perpendicular to the long side of the lever. This requires the deliberate configuration of the robot base position, the balancing angle, the waist and the arm joints.

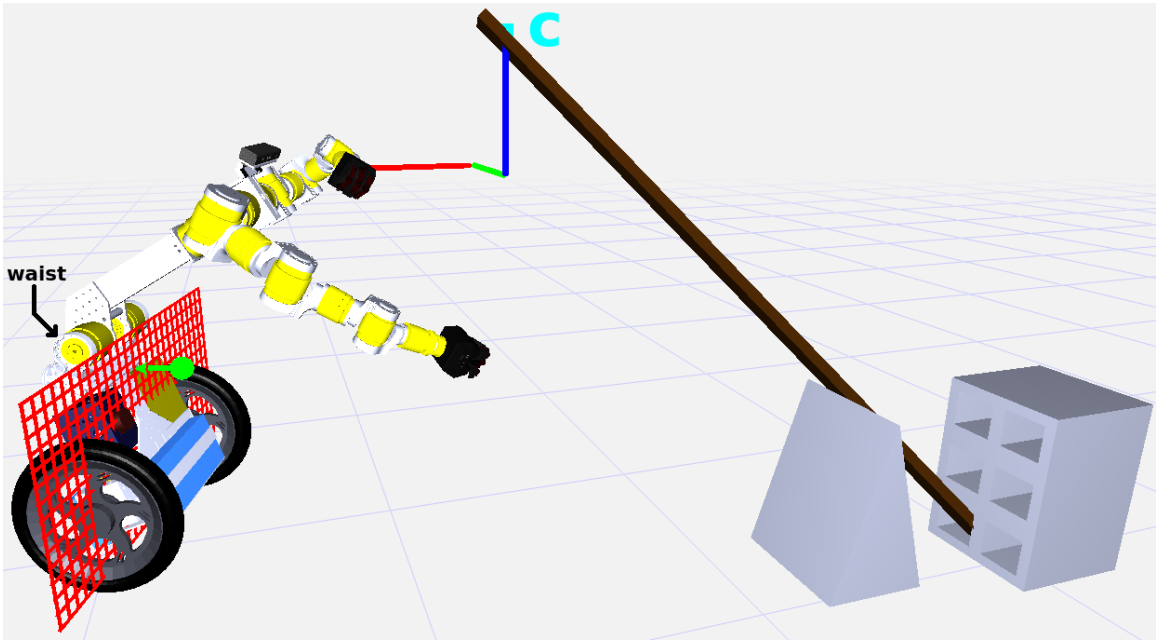


Figure 18: The distance to the desired contact point (cyan) displayed in axis aligned, red, green and blue line segments; and the balancing error shown between the center of mass (green) and the vertical wheel axis plane (red) for the robot Golem Krang.

Secondly, the robot needs to be in a balanced posture before it makes contact with its environment. For a wheeled balancing robot, the center of mass of the system needs to lie on the vertical plane that crosses the wheel axis with some tolerance due to wheel friction and elasticity:

$$\begin{aligned} \text{reachability: } & f_{LH}(q) - c, quat({}_wT_L) = 0_7 \\ \text{balance: } & \left| \frac{\sum_{\lambda \in \Lambda(R)} m_\lambda * (f_\lambda(q))_x}{\sum_{\lambda \in \Lambda(R)} m_\lambda} \right| \leq k_{COM} \end{aligned} \tag{14}$$

where k_{COM} is the tolerance, c is the contact , $f_{LH}(q)$ is the left hand pose, and $\Lambda(R)$ are the robot links used to compute COM in the sagittal plane. Note that ${}_wT_L$

represents the lever pose in the world frame and the $quat(.)$ function returns the axis perpendicular to the long side of the lever.

Torque limits also need to be considered in the design process if the robot is to actuate the system. In this work, the waist and the wheel motors apply torque and the arms are held fixed mechanical breaks for simpler kinematics and safety. To model the force-torque relationship, we assume quasi-static dynamics with $\tau = J^T(q)f$ where the desired torques τ can be extracted through the contact c and the corresponding robot pose q, and the force magnitude f_m. The magnitude is a function of contact point, lever pose and load mass but we omit the specifics due to lack of space. However, we show examples of how similar geometric projections are used in contact modeling next.

6.2 Adapting Planning Definitions for Complex Tasks

The goal of the planner is to choose from a set of available lever-like objects that can be used with a fulcrum to overturn or push an obstacle. To adopt the approach presented in Algorithm 1, we first define the domain literals and actions. The main difference from the convex stacking domain is about the representation of face and edge information. In addition to the object identities such as fulcrum F and lever L, their unique face and edges (i.e. not symmetric) are also regarded as domain objects. Literals such as $FaceL(A)$ and $EdgeF(B)$ assert that face A belongs to the lever object while edge B belongs to the fulcrum respectively. The goal literal is $Overturn$ and the initial state has all the objects unused along with the proper face and edge identifications.

Table 3 describes the domain actions. The $Structure$ action parameterized with the lever and fulcrum choices, as well as the fulcrum base face, lever faces for load and fulcrum and the fulcrum edge constructs an initial lever-fulcrum design without

Table 3: Lever-fulcrum domain: preconditions, after effects, constraint types and references to Figure 19

Action	**Preconditions**	**Effects**	**Constraints**		
			Goal	**Equation Type**	**#**
$Structure(L, F, f_G^F, e_L^F, f_F^L, f_{Lo}^L)$	$\neg Used(F), \neg Used(L), FaceF(f_G^F), EdgeF(e_L^F), FaceL(f_F^L), FaceL(f_{Lo}^L). \neg In(e_L^F, f_G^F), Para(e_L^F, f_G^F)$	$Used(F), Used(L)$	Lever-fulcrum contact	Face-to-edge	1
			Lever-load contact	Face-to-edge	2
			Load-fulcrum-input order	$C_{LLo}^x \leq C_{LF}^x \leq C_i^x$	3-4
			Lever-ground collision	$\forall P_i \in M_L : P_i^z > 0$	5
			Lever-load side collision	$\forall \in M_L : \|\|(P_i^z, \text{wall})\|\| > 0$	6
$Contact(f_R^L)$	$Used(L), FaceL(f_R^L), \neg Perp(f_{Lo}^L, f_R^L)$	$InPlace$	Robot-lever contact	Contact positioning	7
			Kinematic constraints	Reachability and balance	8
			Contact in front of robot	$C_i \cdot heading > 0$	9
$Push()$	$InPlace$	$Overturn$	Torque-load relationship	Jacobian transpose + I.K.	10
			Torque limits	Constant inequalities	11

the robot. Figure 19 demonstrates examples of such assemblies acquired by finding feasible configurations to the contact and collision constraints numbered 1-6. The constraints 3 and 4 ensure that the fulcrum position is between the load and the lever, as opposed to the alternative fulcrum-load-input setup. To limit the possible face-edge matches, the $\neg In(e_L^F, f_G^F)$ and $Para(e_L^F, f_G^F)$ prerequisites are added by the expert user. These literals require that the fulcrum edge for the lever is not on the ground and it is parallel to the ground since the load edge it is connected to assumed to be parallel respectively. Although this symbolic knowledge is not strictly necessary, since the infeasible physical interactions would be rejected eventually, it still plays a significant role in detection of failure states efficiently. We expand on the limiting of face-edge interfaces in Section 6.3.

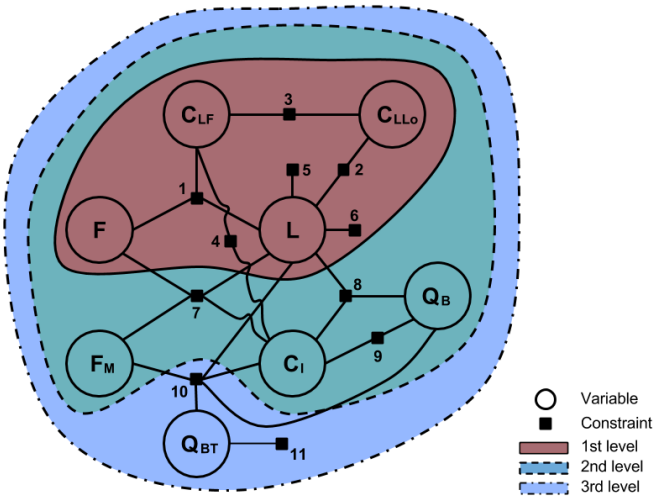


Figure 19: The factor graph that represents the lever-fulcrum design along with the robot configuration.

The *Contact* action, parameterized by the lever face for the input force, incorporates the constraints for the contact point, robot inverse kinematics and balanced posture. If such a configuration is possible, the InPlace literal which is a prerequisite for the *Push* action is added. Finally, *Push* incorporates the torque relationship to the load mass and sets its limits. To find the final configuration in a 26-dimensional

space, the nonlinear feasibility test presented in Algorithm 2 in Section 5.3 first solves for the 1st level of lever-fulcrum assembly alone (red), then incorporates the robot kinematic limitations (green) and finally ensures the robot has sufficient torque (blue).

6.3 Limiting Face-Edge Interfaces with Domain Knowledge

If we assume that the lever object is represented with a rectangular prism mesh with 6 faces and the fulcrum is a triangle prism with 5 faces and 9 edges, the number of combinations of the five parameters to the *AssembleStructure* and *MakeContact* functions is 9720. This large number of *discrete* roles would imply an insurmountable amount of nonlinear optimization time which is already bottlenecked by the random restarts. In this section, we provide several insights that are used to prune the space. Figure 20 represents four distinct examples where only the contact constraints defined in the *AssembleStructure* are used to find feasible configurations.

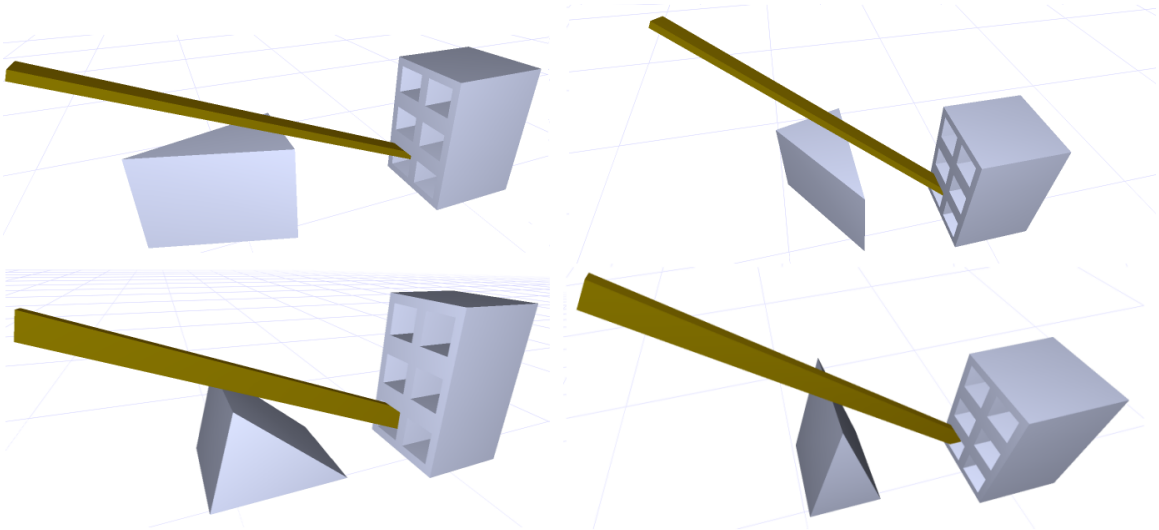


Figure 20: Effect of different face-edge matches between the lever, fulcrum and load objects to design configurations

First, we observe that most of the options are symmetrical. For instance, for a

lever object, it does not matter which long face we use to make contact with the fulcrum. Instead of using the 6 faces for the lever, if we consider only 3; and for the fulcrum if we use 4 edges and 3 faces, by eliminating the symmetries, we end up with $6 \times 6 \times 6 \times 4 \times 3 = 324$ possible configurations. Note that the same face can be used for some purposes - for instance, the load and the robot can push on the same side of the lever. However, some of these combinations are not possible because of the following reasons:

1. fulcrum edge is adjacent to the fulcrum base which places the fulcrum edge on the ground,

2. input or load faces are not perpendicular to the fulcrum face which means that the input or the load can not generate torque around the fulcrum,

3. fulcrum edge is not parallel to the load edge which is necessary for the level to make edge contacts with both the load and the fulcrum - similar concept to the effect of the contact constraints on the orientation of the objects in the ramp example.

With each pruning of a match, the total number of possibilities decreases by two-fold or more, until **fifteen** constraints are left. The decrease in the object role space from 9720 to 15 is the primary reason why object roles are effective in this domain where the objects can be represents by meshes with small number of faces. If that was not the case, even with pruning heuristics, the role space might be intractable.

6.4 Experiments

In this section, we first propose that the physics that underlie the use of a lever-fulcrum system can be approximated with simple quasi-static constraints on the robot and objects pose. Moreover, we claim that using this model, a high-level planner can

choose which lever or fulcrum to choose, and which face or edge they should interact with each other or the ground with. We begin with evaluating our model and how accurately the joint torques and the related end-effector forces are minimized using a lever.

The experimental process is as follows: (1) mesh data for available objects are input, (2) the planner outputs a design, (3) human collaborator places objects and the robot in the instructed configurations, and (4) robot applies the input force. Figures 21a-b demonstrate the output of the planner in a graphical user interface and its human replication in real-world.

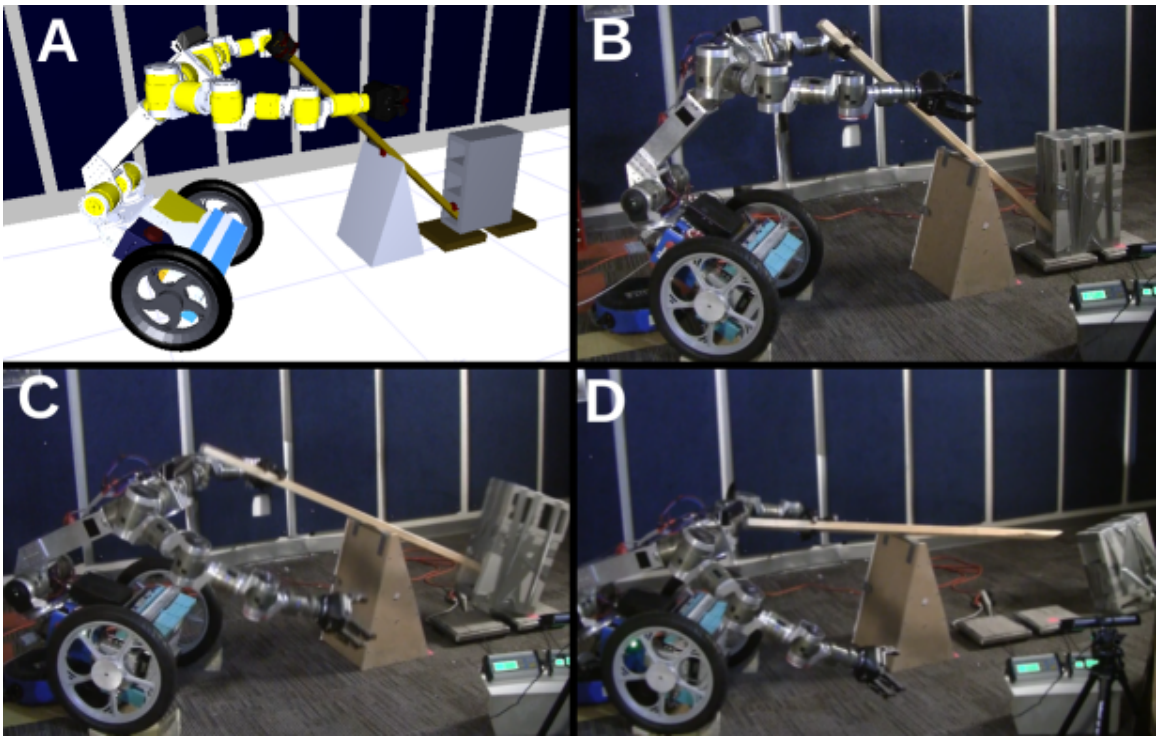


Figure 21: The ideal planner design, replication by human collaborator and key frames from the actuation by Golem Krang for a 50 kg obstacle and 1.7 m lever

6.4.1 Force Analysis in Real Life Experiments

To evaluate the accuracy of the actuation of the design, we have accumulated data from three sources: (1) force/torque sensors at the robot gripper, (2) wheel torque

sensors, (3) scales placed under the loads. In Figure 21b, the two scales can be seen under the load with the camera recording the values. Figure 22 plots the data from the 50 kg overturning experiment where the object finally topples over just before the 90 second mark. Note that we have marked three points of interest shown in Figure 21b-d with gray vertical lines: (1) first application of more than 30 N, (2) the load standing on its back edge and (3) the load falling down.

First, note that the oscillations in z-axis forces and the wheel torques after the fall are due to the robot trying to regain balance while holding the lever. Second, the maximum input force is 205.2 N and the wheel torque is -52 Nm. We propose two reasons why these values are less than expected. First, the planner ignores the forces perpendicular to the axis of rotation of the lever, which can actually help overturn the load. Second, we observe that the waist angle (black, dash-dotted) changes as the input force increases although we assumed the joint angles are fixed. We suspect the reason is the mechanical compliance of the robot, in addition to the flexing of the lever, that causes the upper body to move as opposed to the load. The last observation is the motion of the load center of mass is observed in the decrease in the front scale readings and the corresponding increase in back.

6.4.2 Feasible Component Choice

We now evaluate whether the model, despite its imperfections, can be used for high-level planning where the robot chooses the design components and how they should interact with each other. The scenario is as follows. The planner is provided two lever options with lengths 1.7 m and 2.5 m, and is given two obstacles that weigh 50 kg and 100 kg. In addition, the robot joint torques are constrained to 150Nm.

The first task is to topple over the 50 kg obstacle. Given the heuristic to prioritize the shorter lever, the planner determines that the short one is sufficient to overturn the object using a desired input force of 220 N with (-67, 118) Nm torques and a

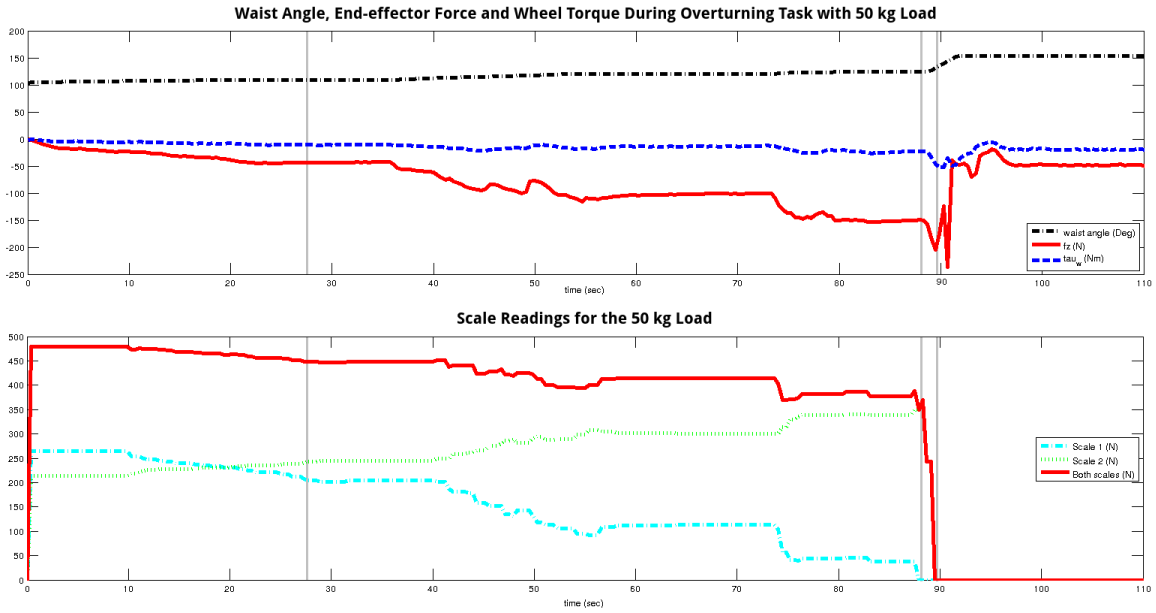


Figure 22: Force and torque readings throughout the 50 kg overturning task. Top graph: input force along z-axis (solid), wheel torque (dashed) and the waist angle (dashed-dotted). Oscillations in force after fall is due to robot's attempt to regain balance. Bottom graph: the decrease of the measured load mass and the shift of its center of mass as shown by the scale measurements.

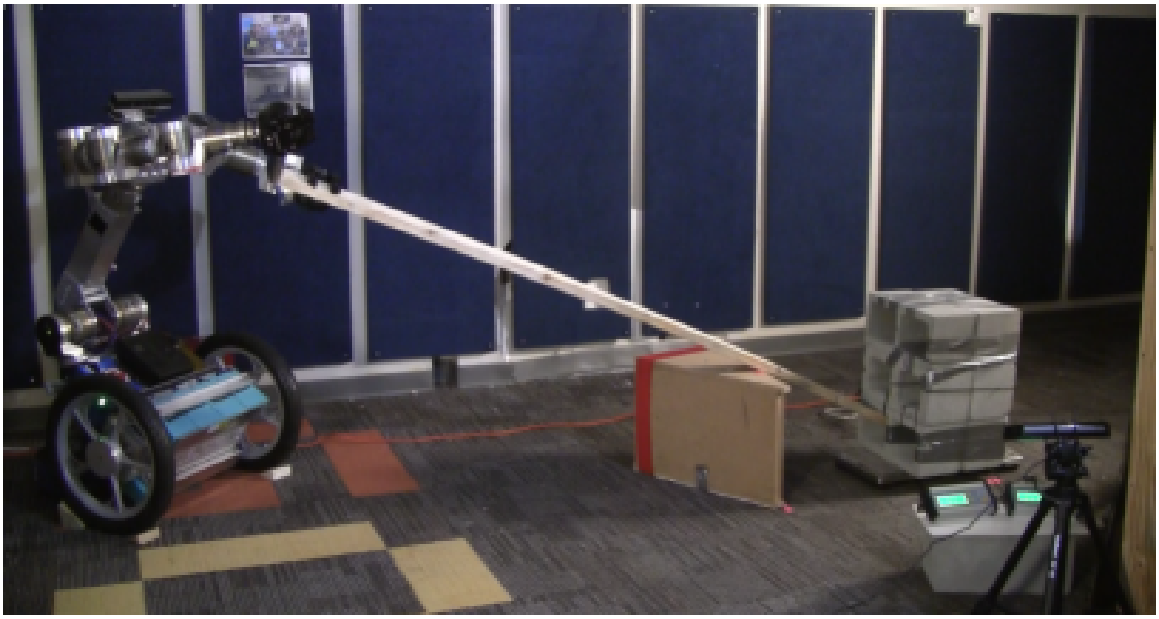


Figure 23: Golem Krang, lever and fulcrum poses to topple over 100 kg obstacle with a 2.5 m lever

mechanical advantage of 1:2.2. However, when the heavier 100 kg obstacle is given, the planner chooses the longer lever shown in Figure 23, using 151 N with (-76, 111) Nm and an advantage of 1:6.50. To topple the heavier object with the shorter lever, the robot would have needed more than 200 Nm at its waist joint, and thus, the feasibility tests successfully dismissed all the 15 different short lever uses.

We also conducted an obstacle pushing experiment where a door is blocked by a 240 kg wheeled platform. The goal of this experiment is more of a demonstration than an evaluation where the insight is that the *same constraints* can be used to model interactions that are parallel to the ground as opposed to perpendicular since constraints are local between the interacting objects. We only had to incorporate a single new constraint which insists that the fulcrum is positioned in contact with the wall to withstand the normal force from pushing the door. Given that the load weight is 240 kg and the ground ground friction, 1500 N is needed to move the platform using a lever. Figure 24 shows Golem Krang opening the door using a planner design.

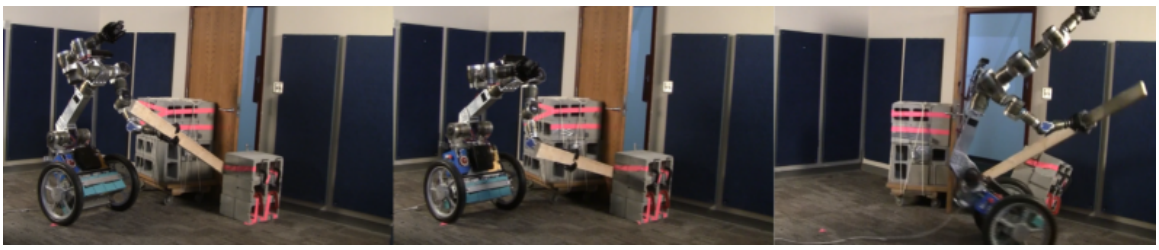


Figure 24: Key motion steps in pushing a heavy object

6.5 Analysis and Conclusion

We begin our analysis with the factors that affect the efficiency of the approach. A challenge in this domain, in addition to nonlinear constraints, is the connectedness of the variables with an average degree of 3 neighbors in Figure 19 compared to the simple ramp structure in Figure 16a. The sparsity of the factor graph can play a direct role in the efficient resolution of the constraints [24]. In addition, the commitment to

initializations and random restarts lead to average *total* optimization times for the domain factor graph in 81.32 and 55.08 seconds for the overturning and pushing tasks over 20 random trials. A major challenge in future work is the analysis of analytical or grid-based optimization approaches that avoid arbitrary initializations in the global minima search. Lastly, we observe that with more challenging tasks where more mechanical advantage is needed and the feasible subspace of assembly configurations is smaller, it is harder to find good initializations that lead to the global minima.

Aside from efficiency, another interesting issue is the use of heuristic collision checks to reduce the motion planning problem to the analysis of a single instant where initial and maximum force is applied to the lever by the robot. The collision constraints are necessary to ensure that the robot is not configured in a pose that would collide with the environment and more importantly, the lever does not hit the robot as it is pushed down. Although this heuristic is successful in limiting the space of configuration to only collision-free ones, it is inevitably conservative. We leave this issue of foreseeing the object use and its incorporation in the design process for future work.

Chapter 7

Autonomous Construction of a Simple Machine

A crucial assumption in the experimental analysis in Chapter 6 is that a proposed design by the planner can be constructed perfectly. The error between the ideal structure and the constructed one was minimized by a human collaborator who had to meticulously account for the placement of the objects and the robot pose. However, in search and rescue operations, the robot that is in need of a functional structure, is also responsible of constructing it. Subsequently, errors in execution inevitable would lead to imperfect structures.

In this section, we present initial work on the autonomous construction of functional structures. The goal is for Golem Krang to autonomously perceive its environment, generate motion plans to manipulate cinder blocks and wooden pieces, and actuate a lever-fulcrum to flip over a 50 kg obstacle. Figure 25 demonstrates the beginning of the experimental setup where Golem Krang searches for candidate fulcrum and lever objects and its completion where the structure is constructed and the robot pushes on the lever.

We begin the discussion with a brief explanation of the prerequisite modules for

Figure 25: Experimental setup: Golem Krang searches its environment for cinder blocks to use as fulcrums and 2x4 wooden pieces as levers.

autonomy, then run through an example walkthrough, and conclude with the insights on how the design process can be improved.

7.1 Preliminary Modules: Perception, Motion Planning and Control

Equipped with a Microsoft Kinect that can pan and tilt, Golem Krang can inspect and reason about its environment with visual data. In perceiving the environment, we propose using a light-weight feasure-based recognition approach as opposed to full 3D based approaches that use the entire mesh data such as the iterative closest point algorithm or over-segmentation methods. An assumption is that the planner knows the meshes of the available objects and with minimal additional feature knowledge, such as the top of a cinder block is at 44 cm from the ground or a lever is at least 2 meters in one dimension, we can speed up the detection. The proposed approach, detects individual and/or assembly of cinder blocks, walls and wooden blocks as shown in Figure 26.

The primary locomotion strategy for Golem Krang is to balance on its two wheels, keeping its center of mass on the vertical plane through its wheel axis. Modeled as an inverted pendulum, locomotion via balancing has a few advantages over running

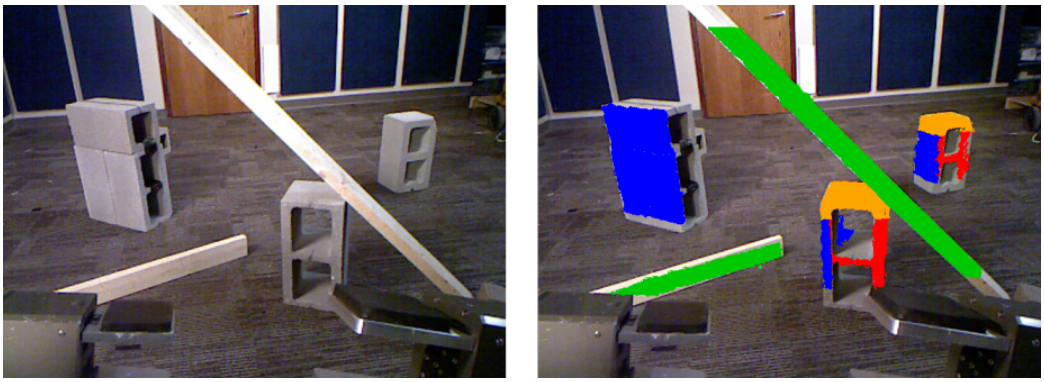


Figure 26: Cinder block and wooden plates detected as fulcrum and lever objects.

on both the wheels and the back caster. First, the footprint of the robot is smaller, 54 cm in width due to the wheel diameter while balancing, as opposed to 86 cm when the robot is grounded. Second, the locomotion is simpler to model since the fixed caster without omnidirectional wheels sustains different ground reaction forces as the robot spins, moves forward and backward.

The position and posture control is implemented using a proportional derivative controller based on the inertial readings that indicate the robot angle from the vertical and the wheel encoders. In this work, we assume the environment is setup such that the locomotion can be carried out by turning towards the goal position, moving forward and adjusting for the goal orientation - ignoring collisions in the world. To move forward, we use a velocity profile with limits on minimum and maximum acceleration and deceleration. A significant aspect of the locomotion is the manipulation of heavy objects such as 15 kg cinder blocks and 10 kg wooden plates. To enable stable dynamic balancing, the force-torque sensors at the grippers are used to incorporate the mass of the carried objects in the computation of the center of mass position.

The manipulation of multiple objects under motor and perception uncertainty requires a series of robust strategies both algorithmically and in practical implementation. In this work, a wide range of motion planning tools are adopted such as rapidly-expanding random trees (RRTs) [78], analytical inverse kinematics [126] and

Jacobian control [135]. Additionally, we propose using guarded moves [6] that control the manipulator behavior until a predetermined tactile feedback is received. Moreover, "conformant motions" are used where the robot forces itself and its environment to a desired state without sensory feedback [49].

Figure 27 depicts the analytical inverse kinematics and the steps during the manipulation of a cinder block to be used as a fulcrum. We use a predetermined grasp location, the top surface with the holes at the sides (see Figure 27a). The planner first uses analytical inverse kinematics to find configurations close to the object that are collision-free. Figure 27a displays three configurations out of which the left most, semi-transparent one collides with a wheel. Once a goal in arm jointspace is determined, bidirectional RRTs with path shortening and smoothing are used to move the arm from its initial pose to the goal. Figure 27b demonstrates the keyframes as the arm moves from its initial pose (red), around the cinder block to avoid collisions, until it reaches the goal pose (green) in front of the grasp point.

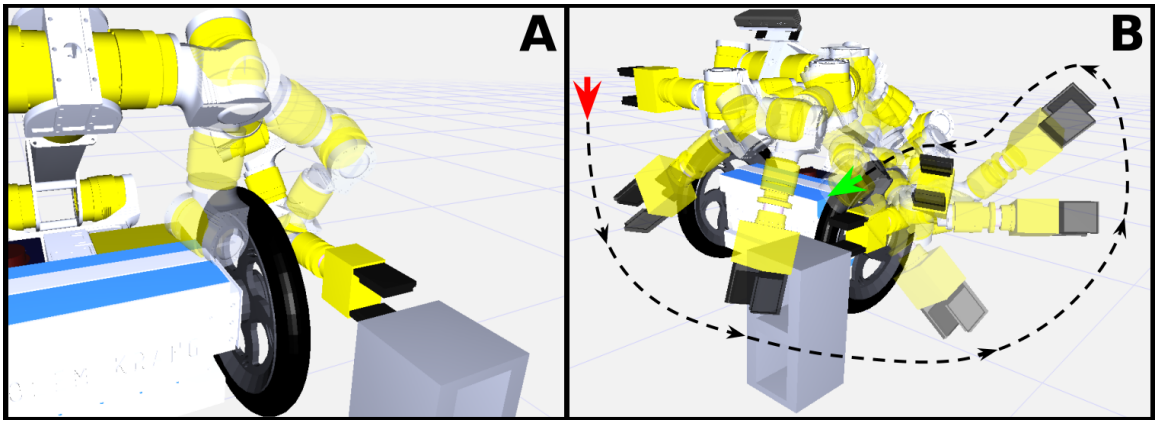


Figure 27: Left: Candidate grasp poses for the block - left most in collision with wheel. Right: RRT trajectory to goal grasp pose, moving around the block to avoid collisions.

Once in position for grasping, Golem Krang uses force/torque feedback at the end-effectors to reach out to the cinder block until contact and ensures its grippers can grasp it. Such guarded moves have proven to be simple, heuristic alternatives to

visual servoing as the robot picks up levers and positions them on the fulcrum and inside the load. We also utilize conformant motions where, to localize the lever more precisely, the robot runs its wheels against the lever. When one of the wheels hit the obstacle first, the other wheel comes around until both wheels are in contact and any visual position error is removed. Such motions are used to eliminate uncertainty in the initial pose of objects in assembly tasks by pushing them into known poses [103].

7.2 Execution Walkthrough

Golem Krang is tasked with overturning a 50 kg load using a lever-fulcrum assembly with a limit of 300 Nm on the force it can apply to the environment. Given the dimensions of the available objects, the robot has to design a structure, locate the components, position them and actuate the simple machine.

Placed in a random configuration in the room, Golem Krang begins by scanning the room for the available objects and finds the closest cinder block that would be used as a fulcrum (see Figure 28). The scanning process is composed of a set of atomic behaviors which move its arms out of its sight to avoid occlusions. Once the fulcrum is located, the robot approaches until it positions itself in a predetermined distance to grasp the object. Using the motion planning tools, such as RRTs and guarded moves, the robot grasps the cinder block at its top.

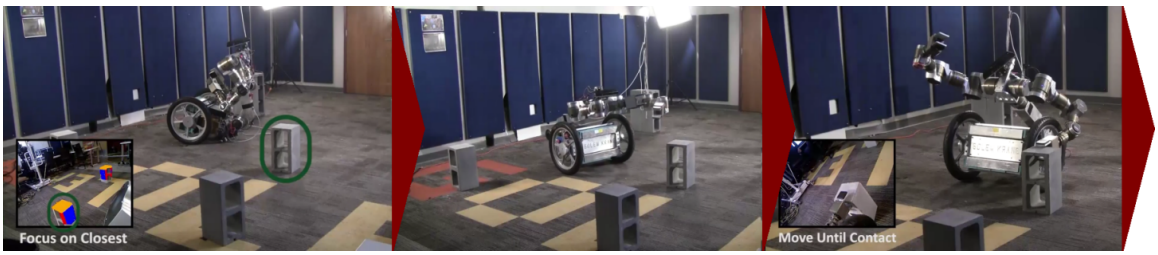


Figure 28: Once Golem Krang detects the closest cinder block (left), it approaches (middle) and grasps the objects (right). Scene continues in Figure 29.

An interesting observation is about how the location of a manipulated object and

the uneven distribution of its weight over the wheels affect the locomotion accuracy. To minimize such an artifact, in Figure 29, Golem Krang moves the grasped cinder block to the middle of its torso before turning around and localizing the load. Having detected the load, the final configuration of the fulcrum is deduced from the assembly design and the robot places it appropriately.

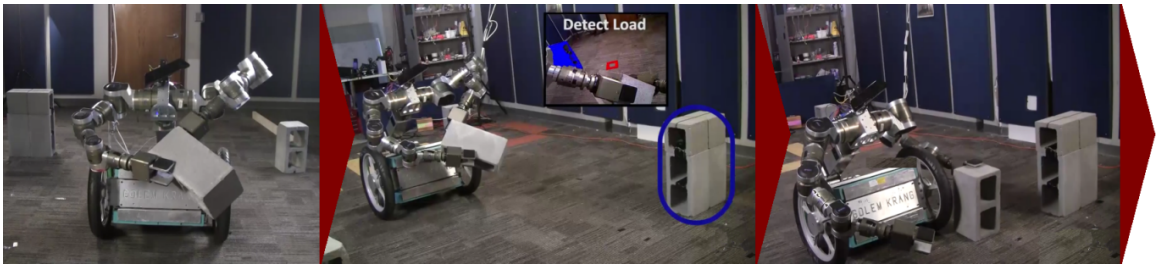


Figure 29: Having grasped the fulcrum, the robot localizes the load and places the fulcrum in the initial design configuration. Scene continues in Figure 30.

In the third part of the experiment, Golem Krang needs to detect and localize a candidate lever object and grasp it, as shown in Figure 30. Given the size of the lever and the noisy perception data, we propose using the wheels to localize the lever object more accurately once the robot approaches it. Figure 30b displays the conformant behavior where the robot moves forward slowly to collide with the lever and have its localization error fixed. The left wheel first makes contact and the contact overcomes the input torque, while the right wheel keeps moving until the robot is parallel and directly in front of the lever.

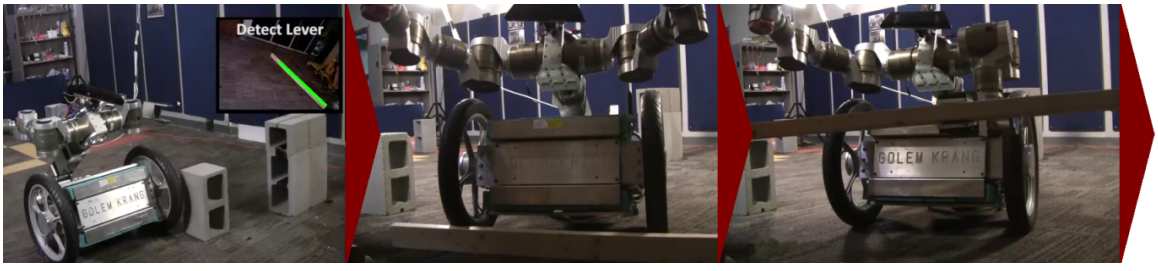


Figure 30: The lever is picked up by first using vision and then running the wheels against the object to make physical contact before manipulation. Scene continues in Figure 31.

To simplify the locomotion, we have assumed collision-free paths and when Golem Krang carries the lever, we ensure that the lever is carried high enough that it does not collide with other objects (see Figure 31). Once the robot repositions itself in front of the fulcrum, using guarded moves, the robot first pushes the lever against the load horizontally to ensure it is at the correct distance and then tilts it until the design angle. When the lever reaches the goal pose, it is released so that it slides on the fulcrum into the load. Finally, the robot pushes the lever at the desired contact point and overturns the load.

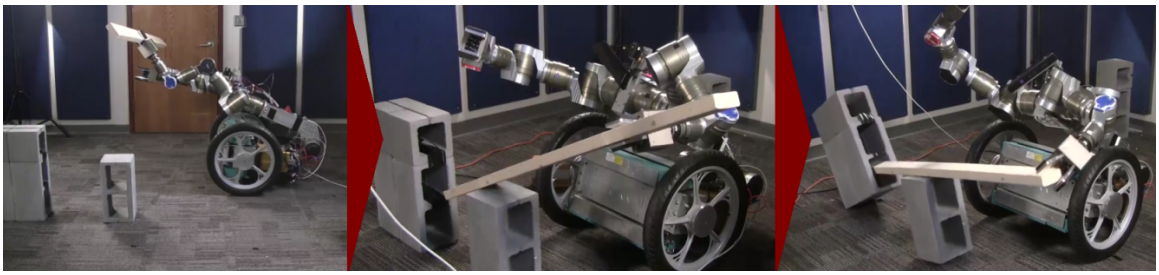


Figure 31: Golem Krang places the lever in the planned pose and overturns the 50 kg load.

7.3 Discussion and Conclusion

The goal of this work was to analyze the effect of discrepancy between an ideal planner design and the output of an autonomous construction. Due to multiple sources of uncertainty in the system such as perception and motion model, we had to teleoperate the robot at times to minimize errors. Even with human oversight, we observed errors 3-5 cm in the final configuration of the components, which in contrast to the planner designs optimized to an accuracy of millimeters, suggest an improvement to the design process.

A possible approach is to incorporate manipulator precision in the constraint optimization framework where the design that can be constructed with minimal error

is preferred. On the other hand, the results bring into question the benefits of reasoning in the continuous configuration space of the objects. *Although continuous assignments is necessary to satisfy the design constraints, a hybrid approach where first the constraint violations are minimized with discretized assignments and then correct assignments made with continuous values might be the correct approach.* The adaptation of design process to execution uncertainty is an open question.

In addition to execution uncertainty, the design process also needs to address the motion planning necessary for the assembly process. As we encoded the atomic behaviors that allowed to robot to pick and place objects, we observed the significance of the robot pose to enable the assembly of the process. In more confined environments, such as those in search and rescue operations, the collisions with the environment would play a more important role on the assembly and therefore, the designs.

Finally, we would like to make a note on the grasping problem that underlies all the manipulation tasks. Clearly, the executed tasks envolve manipulation of heavy objects, in the range of 10-16 kg for pick and place tasks, and up to 200N forces for pushing tasks. The grasp selection plays a critical role in manipulating such heavy objects since in some grasps, the smaller joints cannot generate the high torques necessary to support the weight. In future work, we can incorporate the choice of grasps within the planning framework.

Chapter 8

Heuristics using Configuration Space Volumes

8.1 Introduction

A number of state of the art planning problems such as task and motion planning, assembly planning, task scheduling etc. require decisions on discrete and continuous variables. In most cases, such as the case of task and motion planning, the choices on the discrete task variables lead to additional constraints on the continuous motion variables. Recent work [37, 79, 97] has proposed generating task plans by monitoring the existence of feasible continuous parameters at each decision, but not committing to specific instantiations. In this work, we propose a new domain-independent heuristic to guide the task-level choices by evaluating the possibility of a feasible instantiation at each candidate state.

We are interested in a domain-independent heuristic for a backtracking forward planner where, at each discrete search state, a set of continuous constraints define a feasible configuration subspace. The goal is to find a state that has a goal literal and a feasible assignment to the continuous configuration variables. We propose to measure

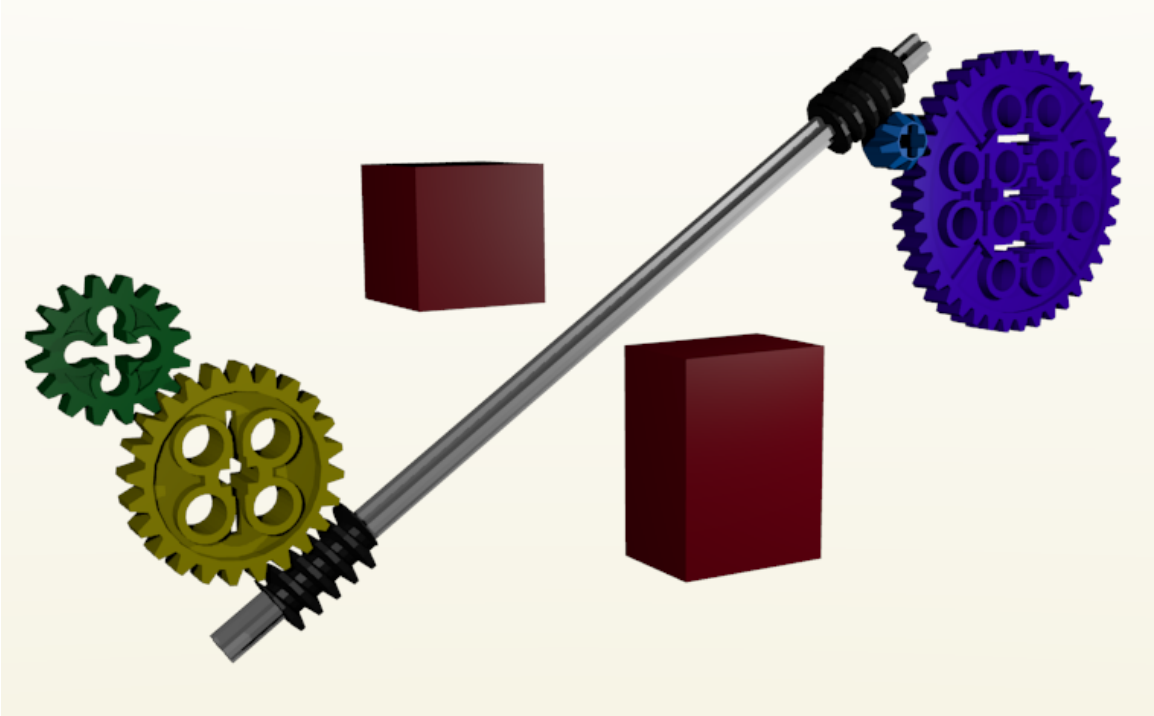


Figure 32: An example of a gear chain designed to transmit mechanical power from the input gear (green) to the output gear (blue) using spur (yellow, cyan) and worm gears (black) and avoiding obstacles (red).

the likelihood of a state being on a goal path by the volume of the feasible subspace defined by its constraints. We show that a state that corresponds to a smaller volume is preferrable because (1) the goal literal and its feasible assignment is more likely to be in a smaller volume state and (2) even if such a state is not a goal state, its children are more likely to be pruned away, limiting the scope of the search.

Despite the extra computation necessary to estimate the feasible space volume, we show that the number of nodes evaluated throughout the search with this domain-independent heuristic is significantly lower to baseline approaches with randomized and hand-crafted heuristics. We demonstrate results both in convex and nonconvex domains. Figure 32 depicts the output of our planner for a nonconvex domain with an assembly task where the inclusion of specific gears and their order are discrete choices and their placements are continuous parameters. The goal is to generate a chain from a motor engine to the output gear, while avoiding obstacles.

We begin the chapter with a discussion of the literature in domain-independent heuristics and volume metrics, and then present the problem definition for the heuristic planning system. Following the presentation of an example domain, we present theoretical justification for prioritizing smaller volumes, and discuss the algorithm. Finally, we show experimental results in stacking and gear domains, and conclude with discussion.

8.2 Related Work

8.2.1 Domain-Independent Heuristics

We present the relevant literature in two parts. First, we give an overview of domain-independent heuristics in a number of AI fields, and then discuss the state of the art in the volume computation of feasible spaces.

8.2.1.1 Constraint satisfaction problems

One of the reasons discrete CSPs stand as powerful abstractions is the domain-independent heuristics in the framework. In discrete CSPs, a planner needs to make two important choices: (1) which variable X_i to include next in a partial assignment, and (2) what value from D_i to assign. The *minimum remaining values* (MRV) heuristic suggests picking the most constrained variable while *least-constraint value* heuristic promotes assigning values in D_i that rule out fewest choices in the other variables' domains [3]. In this work, we propose a generalization of the MRV heuristic to the continuous space where *volume* of the feasible space is used to assess the most constrained search state.

8.2.1.2 Classical planning

Domain-independent heuristics have also been studied as powerful tools in search-based planners [28, 123]. For instance, the prominent FF [61] and GraphPlan [10] planners first operate on relaxed versions of the problem by ignoring some of the action effects (e.g. delete effects). The cost of the optimal plan for the relaxed problem then can be used as a lower bound heuristic for the original problem. The popularity and the success of these planners [93] with powerful domain-independent heuristics motivate the need for domains rooted also in continuous spaces.

8.2.1.3 Operations Research

The domain-independent analysis of discrete and continuous problem spaces (e.g. optimization, feasibility, statististics, etc.) is one of the main concerns of the operations research field. Subsequently, a number of generic heuristics have been developed that lead to significant efficiency gains. One of the earliest examples is the mixed-integer linear programming (MILP) problem where a "pseudo-cost" is generated to evaluate the importance of an integer solution to a subproblem and used as a guide to search for the optimal value [7]. Recently, [9] has proposed to solve a set covering problem to determine which variables to fix in order to linearize the constraints and showed that the solution to the simpler MILP problem also satisfies the original problem constraints.

An interesting subfield of operations research focuses on the minimization of risk under uncertainty in design planning [27, 53]. The approach is to approximate the feasible space defined by nonconvex constraints with a simplicial convex hull and then reformulate the problem as a linear program. Despite the "locally convex" assumption of the constraints, this line of thought is one of the first examples that proposes using the volume of the approximated space as an evaluation metric.

Although volume computation is difficult, we believe the information generated in

the process of finding a feasible solution can be exploited to mitigate its challenges. For instance, in convex domains, a subset of the vertices of the feasible polytope has to be traversed to find the optimal instantiation [21], and these very vertices can then be used to evaluate the volume. In nonconvex domains, once a feasible solution is attained after an innegligible amount of computation, if the volume of the neighborhood of the solution can be efficiently assessed, similar to [27], it can be used as a lower bound on the volume of the entire feasible space.

8.2.1.4 Task and motion planning

The evaluation of constraints in the space of discrete and continuous choices is ubiquituous in planning and optimization problems. One of the earlier domain examples is that of automated space planning [16, 33, 105]. In 1973, Eastman observes that in space planning user studies, "their [users'] treatment of problems is most easily understood in terms of constraints," as opposed to explicitly attempting to maximize a utility function. This observation is paralleled by the constraint satisfaction centered approaches to the mixed-integer planning problems where a number of domain-independent heuristics, including MRV, are proposed to evaluate and exploit the properties of the constraints [47].

In the domain of task and motion planning, the key challenge is the tight coupling between the choices for the high-level discrete task space and the low-level motion plans. Regardless of the underlying sampling-based motion planner (e.g. PRMs [59], RRTs [124]), most approaches face the so-called *commitment problem* where the investigation of a task decision has to be preceded by committing to a low-level motion plan, and any error in that high-dimensional plan may lead to backtracking far deeper in the search tree. Recently Lagriffoul et. al proposed an alternative approach where, instead of finding a specific motion plan, motion bounds are propagated as task choices (and corresponding constraints) are accumulated [79]. The method, also

adopted by [37, 97], enables the planner to first create a *plan skeleton* and then solve a continuous CSP to instantiate the motion. In fact, recent work from a number of diverse fields, from chemical engineering [45] to architectural design [141], follow a similar approach, where first high-level discrete choices are made and then their resulting constraints are solved to instantiate the continuous parameters.

Given the wide variety of design problems and the similarity of solution approaches, it is desirable to formulate a domain-independent heuristic that regulates the search process to prioritize the generation of continuous CSPs that are more likely to have feasible assignments. To this end, we propose using the volume of the feasible space in a task state to guide the planner in making the decision of expanding that state. Figure 33 demonstrates this key idea in our work where the states with smaller feasible subspaces (3) should be prioritized over alternatives (2) because they lead either to goal states (4) or to states with conflicting constraints (5) that are subsequently pruned.

In addition to volume, a number of different heuristics adopted in the MINLP can be incorporated into the TAMP domain and to the best knowledge of the authors, the exploitation of such operations research methods in this domain is not prevalent yet.

8.2.2 Volume of Feasible Space

In this section, we provide a brief overview of the types of approaches used to evaluate the volume of a feasible space defined by a set of (non)convex constraints in algebraic geometry and mathematical optimization. Note that we are interested in the solutions to the constraints that constitute continuous subspaces, possibly disjoint, and ignore point solutions since the volume metric would be inadmissable for these cases.

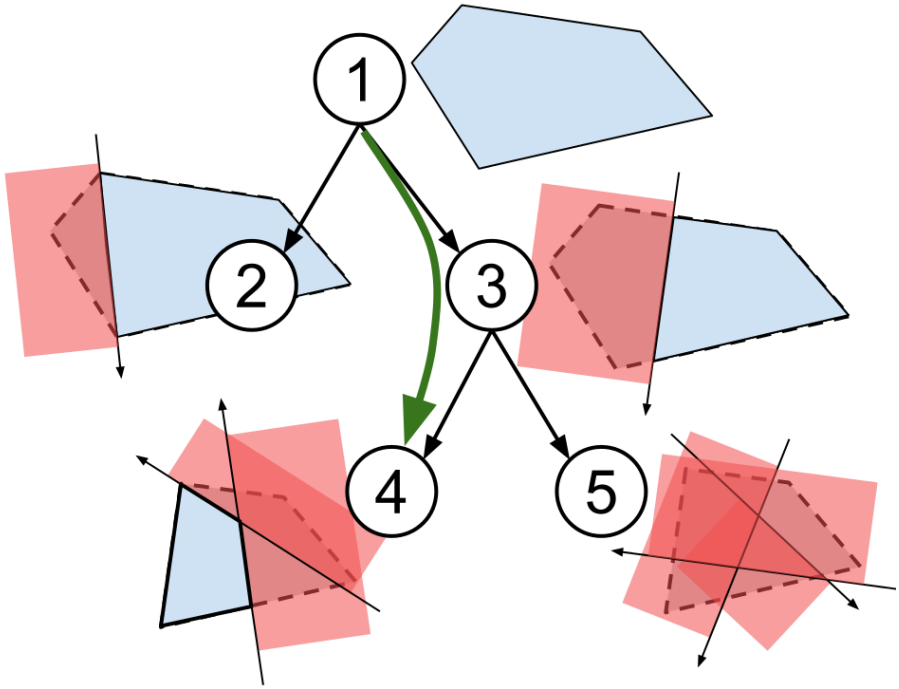


Figure 33: At each state, new constraints are induced, invalidating half-spaces (red) of the initial feasible space (blue). We propose prioritizing states with smaller feasible subspaces while pruning those with none.

8.2.2.1 Volumes of Convex Polytopes

For a set of convex constraints, the feasible space is defined by a *polytope*, the generalization of a polyhedron to higher dimensions. The generic approach to compute the volume of a polytope is through triangularization where the polytope is represented as a set of disjoint simplices (higher-dimensional triangles) [86, 134]. Note that the volume of a simplex can be computed in closed form [60].

The challenge of computing the volume of a polytope increases with the number of constraints and the dimensionality of the space. To generate a triangulation, first the vertices of the polytope need to be enumerated, which is a NP-hard task [1]. However, increasingly more efficient randomized schemes have been proposed to generate approximate enumerations [70, 94, 95].

Even after the vertices are enumerated, the computation of the Delaunay triangulation, or its dual Voronoi diagram can be computationally cumbersome in higher dimensions [11, 13]. However, Emiris and Fisikopoulos, recently proposed an efficient approximation scheme for the volume directly [35]. In this work, we adopt the classical triangulation scheme as a proof of concept, implemented in the Parma Polyhedra Library [4].

8.2.2.2 Approximation of Nonconvex Subspaces

Along with the advantages of nonconvex constraints in modeling complex designs, motions, etc., the computation of the volume of their corresponding feasible spaces become equivalently challenging. As a result, most of the state of the art is still limited to lower dimensional spaces or are theoretical. However, linearization-relaxation techniques that generate outer bounds to the feasible spaces exist [79] and can help generate higher limits on the volumes as heuristics. In addition to relaxation, if a nonconvex polytope can be attained, a constrained Delaunay triangulation [18] of the polytope can be generated to attain a simplicial decomposition. However, the generation of the nonconvex polytope requires sampling in more than 2 dimensions, and even then, new points may need to be generated in the triangulation process [118]. Finally, the simplicial approximation method where a convex polytope is "grown" inside the feasible subspace is proposed by [27]. To the best knowledge of the authors, a generalization of this approach beyond their "locally convex" assumption does not yet exist, although it might be possible through solving a series of recursive, smaller dimension nonlinear programs at each extension of the hull.

Despite the efficiency challenges of the volume metric in nonconvex spaces, we demonstrate the efficacy of this heuristic in decreasing the number of searched nodes in a simple machine design domain in the results section. Within the context of this work, we use exact volume evaluation through sampling, and in future work, we hope

to show that relaxation approximations can function as useful heuristic metrics.

8.3 Foundations for the Volume Heuristic

8.3.1 Goal bias towards smaller subspaces

Theorem 1. Let P be an arbitrary convex polytope in $\mathcal{R}^d$ defined by $m \geq 2d$ linear inequalities $\mathcal{I}$, of which the first $2d$ constitute a bounding box. Let I_{m+1} be a new linear inequality, such that $I_{m+1} \cdot x \leq 0$ for some $x \in \mathcal{R}^d$ constitutes a half-space in $\mathcal{R}^d$. Assume that the intersection of the half-space and P is not empty, and let it be the polytope P'. If I_{m+1} is sampled from a uniform distribution $U_d(-1, 1)$, then the expected value of the ratio of the volumes of P' and P is: $\frac{1}{2}$:

$$\frac{\mathrm{E}(\mathrm{vol}(P'))}{\mathrm{vol}(P)} = \frac{1}{2} \tag{15}$$

Proof. Let $I'_{m+1} = -I_{m+1}$ and let P'' be the intersection of P and half-space of I'_{m+1}. Note that $\mathrm{vol}(P) - \mathrm{vol}(P') = \mathrm{vol}(P'')$. If $\mathbf{I}_d$ is the infinite set of all inequality functions in d-dimensions, we can partition it into two sets based on whether the half-spaces include the origin or not (e.g. constant is less than 0): $\mathbf{I}_d = \mathbf{I}_d^+ \cup \mathbf{I}_d^-$. Note that for each $I \in \mathbf{I}_d^+$, there exists a $I' \in \mathbf{I}_d^-$ such that $I' = -I$, and vice-versa. Let $\mathrm{h}(I)$ denote

the half-space of an inequality. Now, we inspect the expected value:

$$\mathrm{E}(\mathrm{vol}(P')) = \int_{I \in \mathbf{I}_d} \mathrm{P}_{U_d}(dI) \int_{x \in P \cap h(I)} dx \tag{16}$$

$$= \int_{I \in \mathbf{I}_d^+} \mathrm{P}_{U_d}(dI) \int_{x \in P \cap h(I)} dx + \int_{I \in \mathbf{I}_d^-} \mathrm{P}_{U_d}(dI) \int_{x \in P \cap h(I)} dx \tag{17}$$

$$= \int_{I \in \mathbf{I}_d^+} \mathrm{P}_{U_d}(dI) \int_{x \in P \cap h(I)} dx + \int_{I \in \mathbf{I}_d^+} \mathrm{P}_{U_d}(dI) \int_{x \in P \cap h(-I)} dx \tag{18}$$

$$= \int_{I \in \mathbf{I}_d^+} \mathrm{P}_{U_d}(dI) \left(\int_{x \in P \cap h(I)} dx + \int_{x \in P \cap h(-I)} dx \right) \tag{19}$$

$$= \int_{I \in \mathbf{I}_d^+} \mathrm{P}_{U_d}(dI) \int_{x \in P} dx \tag{20}$$

$$= \mathrm{vol}(P) \int_{I \in \mathbf{I}_d^+} \mathrm{P}_{U_d}(dI) = \frac{\mathrm{vol}(P)}{2} \tag{21}$$

Thus, the ratio of the $\mathrm{E}(\mathrm{vol}(P'))$ to $vol(P)$ is $\frac{1}{2}$. □

Figure 34 visualizes the key idea behind Theorem 2, where for each inequality and its half-space intersection, another inequality in the opposite direction and its intersection is shown. Since the volumes of the half-spaces add up to the volume of the entire polytope (areas/polygon in $\mathcal{R}^2$), in average, we can expect half of the polygon to be removed.

Corollary. *Let $\mathcal{P} = \{P_1, P_2 \dots P_n\}$ be a sequence of polytopes defined by a linear inequality set $\{\mathcal{I}_1, \mathcal{I}_2, \dots \mathcal{I}_n\}$ and each set to be generated with the addition of a uniformly sampled random inequality I_i, $\mathcal{I}_{i+1} = \mathcal{I}_{i+1} \cup I_i$, except P_1 which is defined by a bounding box in $\mathcal{R}^d$. Then, the expected volume ratio of P_i with respect to P_1 decreases exponentially:*

$$\frac{\mathrm{E}(\mathrm{vol}(P_n))}{\mathrm{vol}(P_1)} = \left(\frac{1}{2}\right)^n \tag{22}$$

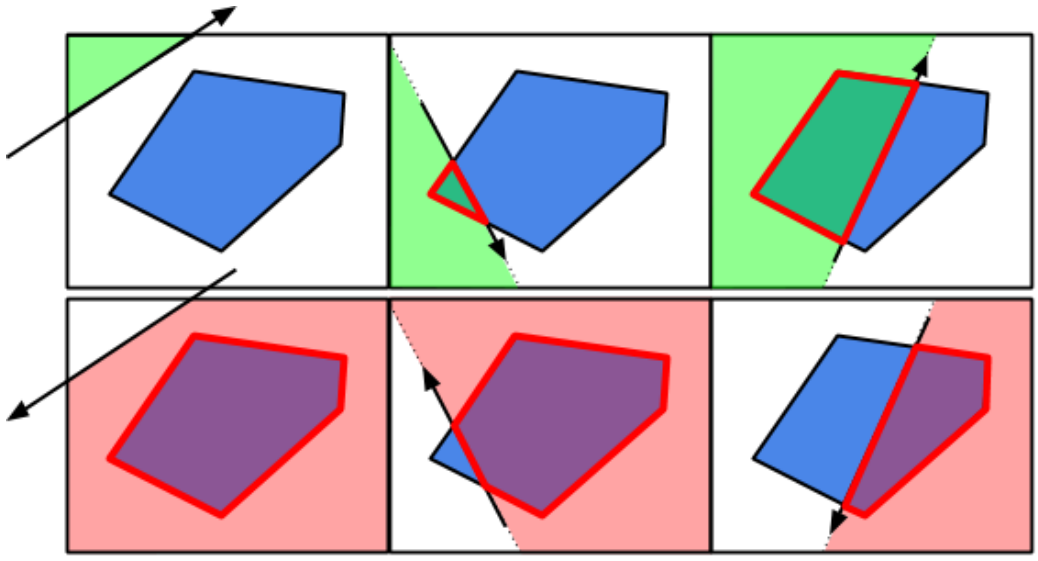


Figure 34: Examples of intersections of half-space and polytope P in 2D

This corollary leads us to the first motivation for adopting a heuristic that prioritizes states with smaller feasible subspaces. ***For non-trivial problems where the solution design requires multiple interaction primitives, the states with smaller feasible subspaces are more likely to be goal states.***

8.3.2 Pruning opportunity in smaller subspaces

Theorem 2. Let $\mathcal{I}$ be a set of linear inequalities in $\mathcal{R}^d$ and let $\mathbb{I}$ be the proper power set of $\mathcal{I}$: $\mathbb{I} = \mathbb{P}(\mathcal{I}) - \mathcal{I}$. Let the notation $X \otimes Y$ indicate the event that the union of two inequality sets X and Y leads to an empty feasible space, which is a conflict. If $I_0 \notin \mathcal{I}$ is an arbitrary linear inequality, then its probability of conflict with $\mathcal{I}$ is higher than that with $\mathcal{I}'$ for any $\mathcal{I}' \in \mathbb{I}$:

$$\mathrm{P}(\mathcal{I} \otimes I_0) \geq \mathrm{P}(\mathcal{I}' \otimes I_0) \quad \forall \mathcal{I}' \in \mathbb{I} \tag{23}$$

Proof. The event $\mathcal{I} \otimes I_0$ can take place in two distinct ways. First, a subset of $\mathcal{I}$ that does not include any element of $\mathcal{I}'$, say $\tilde{\mathcal{I}}$ might conflict with I_0. Second, a subset of $\mathcal{I}'$ might conflict with I_0, also satisfying the event after the inequality. Thus, the

probability $\mathrm{P}(\mathcal{I} \otimes I_0)$ can be decomposed as:

$$\mathrm{P}(\mathcal{I} \otimes I_0) = \mathrm{P}(\tilde{\mathcal{I}} \otimes I_0) + \mathrm{P}(\mathcal{I}' \otimes I_0)$$
$$\forall \tilde{\mathcal{I}} \subset \mathcal{I} \text{ s.t. } \tilde{\mathcal{I}} \cap \mathcal{I}' = \emptyset \tag{24}$$

which is clearly greater than or equal to $\mathrm{P}(\mathcal{I}' \otimes I_0)$. □

To demonstrate the idea behind Theorem 3, we performed an experiment where, at each trial, a number of randomly generated linear inequalities are accumulated in a system until a feasible solution cannot be found. The system is instantiated with a bounding box and the simplex algorithm [22] from the glpsol library [101] is used to find a feasible assignment of values in $\mathcal{R}^2$. The inequalities are generated from a uniform distribution $U_2(-1, 1)$. Figure 35 presents the histogram and the cumulative histogram of the number of inequalities that were needed for the feasible space to vanish in 450,000 trials. The cumulative histogram demonstrates that the likelihood of a set of randomly generated inequalities leading to a conflict increases as the number of inequalities increases.

From the planner stand of view, Theorem 3 establishes the idea that states at deeper levels in the search tree are more likely to be pruned. Thus, we can establish a second motivation to prioritize the states with smaller feasible subspaces. ***If a state is selected due to its smaller feasible subspace but does not constitute a goal state, it is still beneficial to analyze it to prune the tree and limit the search space.***

8.4 Algorithm

Given the planning domain definition and the heuristic of choosing the states with smaller volumes of feasible subspaces, we can now define the planning algorithm as follows. Although there are a number of classical planners with heuristics, such as

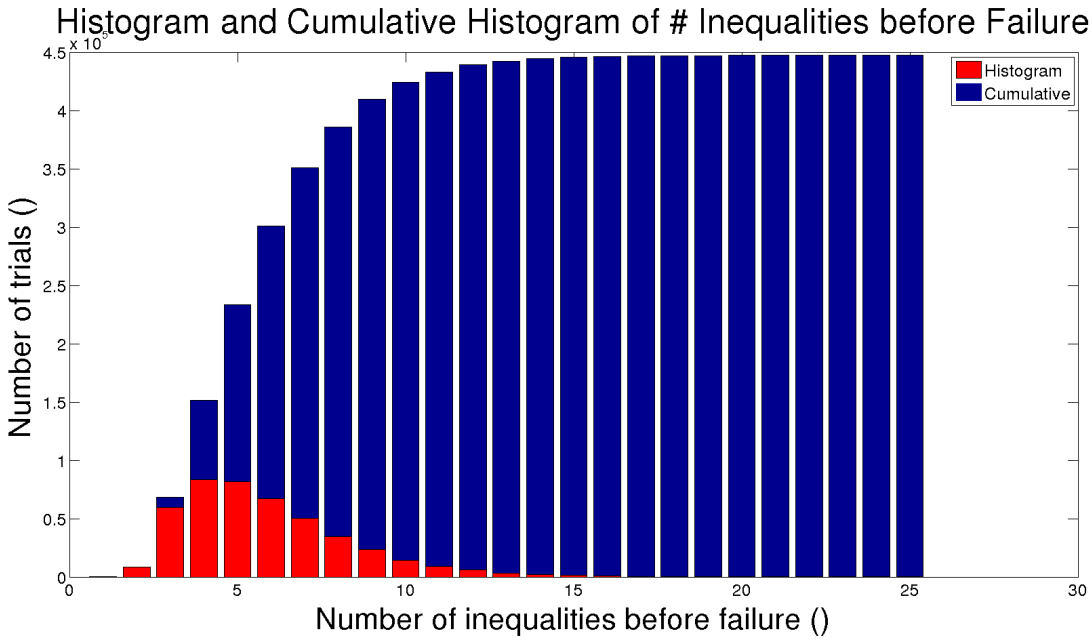


Figure 35: The relationship between the number of randomly generated inequalities in a system and the conflict likelihood in $\mathcal{R}^2$.

A*, as a proof of concept, we deploy the simplest best-first heuristic search, where at each iteration, the state with the higher heuristic value (i.e. the smallest feasible volume) is evaluated.

Algorithm 1 below delineates the pseudo-code for our approach where a priority queue with volumes as the keys is used to determine the evaluation order of the states (lines 1-2). The evaluation of the state involves first checking for the goal literals (line 5) and then ensuring the feasibility of the accumulated constraints (lines 7-8). The volume is computed at line 12 as the newly formed states are pushed into the queue.

8.4.0.1 Volume computation

For the convex domains, we have first used an exact vertex enumeration of the polytope defined by the constraints, then generated a convex hull of the vertices, and finally triangulated the hull to compute the volume of the space. For nonconvex domains, we adopt a rejection sampling method and compare the use the ratio of the number of acceptances and rejections, out of a total set of $N = 1e^5$, as the volume

Algorithm 3: HeuristicConstraintPlanner()

Input: $\mathbb{A}$: domain actions, $\mathbb{O}$: domain objects,
$\mathbb{P}_R$: robot properties, $\mathbb{P}_O$: object properties,
S^0: initial state, l^g: goal literal
Result: X: functional assembly configuration

```
queue ←createPriorityQueue(S^0);
while state ← queue.pop() do
    actions ←instantiate(𝔸, state, 𝕆, ℙ_O, ℙ_R);
    foreach A_i in the set actions do
        if L_i^p ⊄ state.literals then continue;
        C^new ← state.C ∪ C_i;
        X ← feasibilityTest(C^new, state)
        if X = ∅ then continue;
        else if l_g ⊂ A_i.L_i^a then return X;
        else
            child = {state.literals ∪ A_i.L_i^a, C^new};
            queue.push(child, volume(C^new));
            break;
return ∅;
```

indicator of the initial bounding box.

8.5 Experimental Results

8.5.1 Convex Stacking Domain

The results were generated in the stacking domain where for each experiment, the obstacle was set at a height between 0.5 and 0.9 meters, and the dimensions of 2-4 random objects were sampled uniformly between widths 0.2 and 0.7 meters and heights 0.1 and 0.35 for the planner to use. The maximum step height of the robot is set to 0.25 meters. The planner attempts to generate an assembly that the robot can traverse to reach the top of the obstacle, using either a random heuristic, our volume heuristic or a deterministic approach. The deterministic approach would consist of prioritizing the Jump actions, and delaying Put actions so that the assembly

has minimal set of objects. Moreover, the determinism chooses objects with larger surface areas over smaller ones to avoid increasing the search depth by stacking a large number of small items.

Figure 36 demonstrates the log of the number of expanded nodes in the search process averaged among 75 random trials for each height and number of available objects. The volume heuristic (in green) has a lower average consistently across experiments while unexpectedly, the deterministic heuristic is often surprassed by the randomized algorithm.

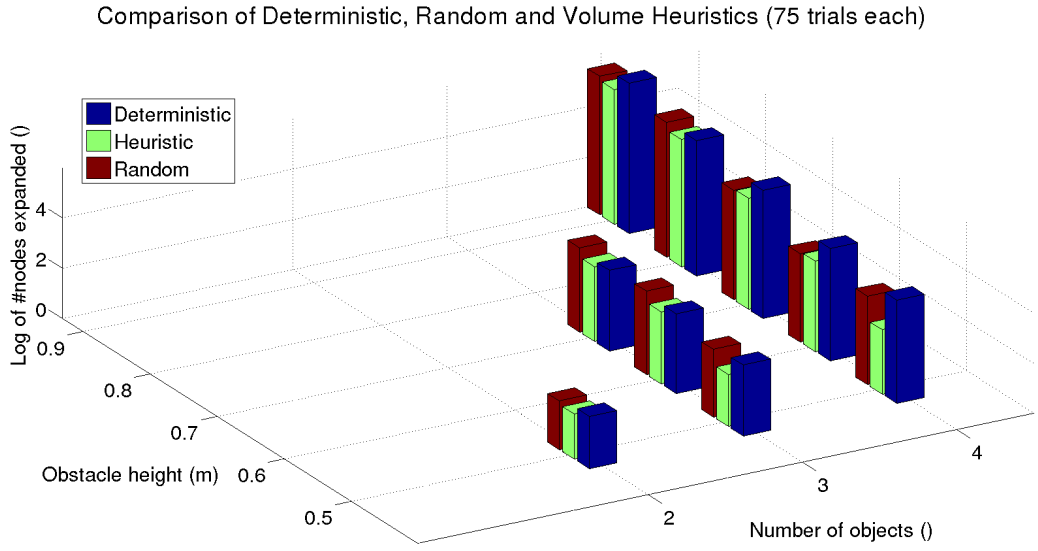


Figure 36: The comparison of number of search nodes between greedy, random and volume heuristics for 2-4 objects with increasing obstacle heights.

One of the reasons behind the effectiveness of the volume heuristic in this domain is its ability to capture the balancing dynamics, the effect of mass and center of mass criteria. Note that the generation of hand-crafted heuristics that account for the details of the wide range of stacked objects topologies would be very challenging.

8.5.2 Nonconvex Gears Domain

In this section, we provide quantitative results on the use of volume heuristic in the nonconvex gear domain. First, note that the domain is composed mainly of quadratic equalities that function as distance constraints and quadratic inequalities that function as obstacle constraints. In our implementation, we relax the equalities as inequalities to preserve the dimensionality of the problem space as the task choices are made.

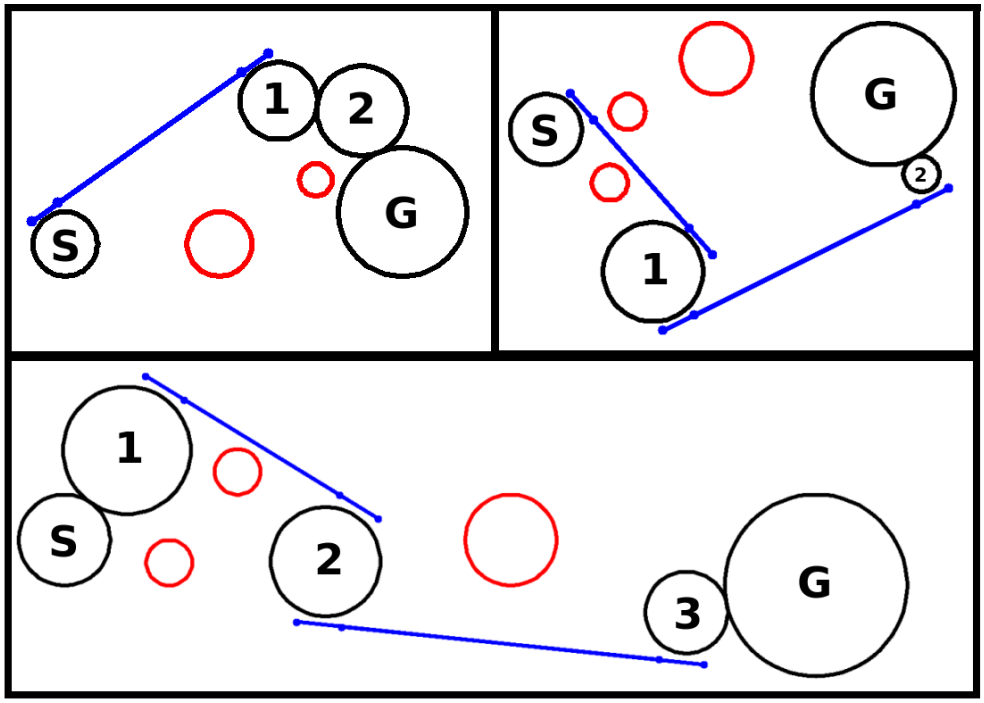


Figure 37: Three examples of gear transmissions from the start gear S to goal gear G while avoiding the obstacles (red). Blue lines represent the worm gears.

The problem has been modeled with four discrete task choices: (1) sequence of spur gears, (2) the connection type of spurs (touching vs. worm gear), (3) sizes of used worm gears, and (4) sizes of the used spur gears. Figure 37 demonstrates the input conditions, that are the the locations of the start and goal gears, S and G, and the obstacles (red), as well as the volume-heuristic planner placements of the spur gears (1-3) and worm gears (blue). Note that although spur gear radii and worm gear

lengths seem to be good resources for handcrafting heuristics, given the wide range of different collision and input gear locations, it is not clear whether such a heuristic would be consistently helpful.

Figure 38 below displays the number of nodes expanded in four consecutively more difficult problem cases. In the i^{th} case, the planner could choose from i spur gears and i worm gears, leading to large decision trees with approximately $4i$ branches in the root node. The distance between the start and goal nodes in $x - axis$ was $\{13, 17, 23, 27\}$ in each case (40 trials).

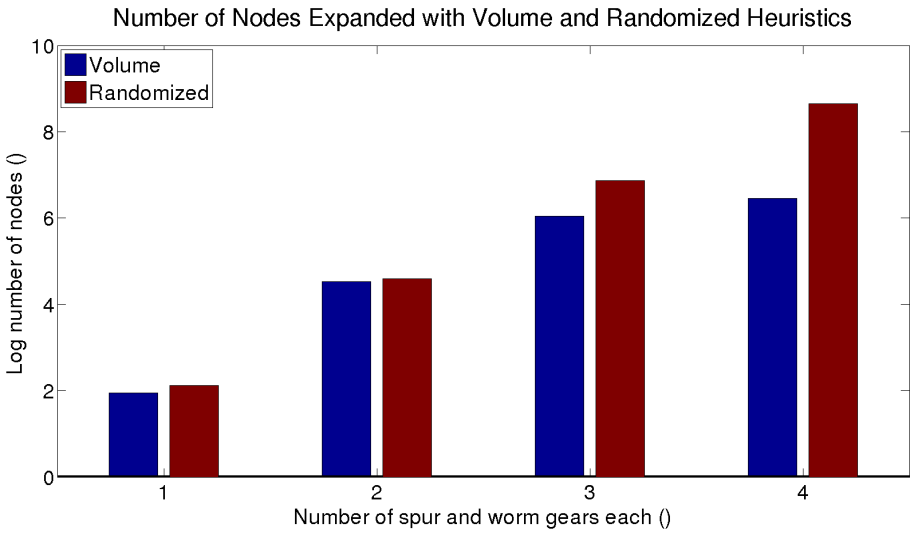


Figure 38: Number of nodes expanded in four cases with increasing number of available spur and worm gears

Note that we make an essential assumption that the volume approximation through sampling is accurate and the results do not take into account the additional efficiency overhead of these computations. However, in future work, we plan to improve upon the simplistic rejection sampling with either MCMC random walks or relaxation techniques [53]. Finally, despite the evident success of the volume metric in these examples, the randomized approach may overtake if the domain example is appropriately designed.

8.6 Conclusion

In this section, we present a domain-independent heuristic to make task-level choices in planning domains with discrete and continuous variables. The proposed volume heuristic prioritizes the states with smaller feasible subspaces for their continuous variables. We are motivated first by work in operations research and constraint satisfaction where a number of domain-independent heuristics are developed. Second, we make the observation that in the process of checking the feasibility of a state, additional information is created that can be exploited towards evaluating the potential of the future states. We present results in two assembly domains where the stacking examples are comprised of convex constraints whereas the gear systems are nonconvex.

In future work, a first challenge would be to address the efficient computation of the feasible subspace volume. As our results show, although the volume metric is powerful, its scalability is a major challenge. Moreover, we propose exploiting the recomputation of feasibility over the search path, where information in previous states can also be put into use.

Chapter 9

Conclusion

The focus of this thesis work has been the design of functional structures to aide robots overcome their physical challenges particularly in search and rescue operations. The key insight that motivates this line of research is that most human environments are rich with resources that can be manipulated and repurposed towards extending robot capabilities. The underlying research challenge has been the "commitment problem" where the discrete choices over component roles in an assembly and the continuous choices over their pose assignments affect the feasibility of one another and henceworth, their interdepency leads to an intractable search space for real world applications. To address this challenge, we have followed a line of research based on the following premise:

> When robots face tasks that challenge their physical capabilities, they can use **constraint satisfaction algorithms** within a **classical planning framework** to create **functional structures** that incorporate multiple objects.

The premise is composed of three essential ideas. First, a classical (i.e. symbolic) planning framework is necessary for imposing sequential construction and use rules for a structure. Second, instead of resolving the interdepency between discrete and

continuous values by enlarging the search space and *committing* to specific continuous poses, we can keep track of the constraints that bound the continuous values and solve a constraint satisfaction problem only for feasibility check purposes. In this manner, the search space is limited to only discrete choices and thus, the computational power can be directed towards solving more challenging problems. Third, the component-component and component-robot interactions that underlie the functionality of an assembly may be reliably expressed such that simulation and real-world results of statis structures and simple machines can be obtained.

9.1 Main Contributions

In [37], we initially propose the concept of lazy commitment and the constraint-based representation of functional structures. The validation results are based on the stacking domain where objects are placed on top of each other with the step size limits of the robot taken into account such that the robot can reach a partition of its configuration space that was not excessible beforehand. The validation results are akin to stairs and overhanging bridges.

In [41], we extend our validation with additional analysis of the effect of heuristics in computation time and number of search states explored, as well as a new domain. The additional goal is to show that the proposed framework can be used for structures beyond 2D and linear stacking actions but with 3D contact mechanisms. Incorporating a symmetry analysis of objects and combinatorial search over face-edge and face-face contacts, the proposed algorithm resolves the placement of ramps and block objects that have higher dimensional and nonconvex interaction principles. Note that while the former body of work, [37], exploits the convexity of the structure constraints with the Simplex algorithm, the latter handles general nonconvex constraints using the Levenberg-Marquardt algorithm.

Having shown the viability of generating static structures similar to stairs, bridges and ramps, in [40], we demonstrate the generation of quasi-static simple machine structures, such as lever-fulcrum systems, where one or more components may move during the manipulation of a payload. In addition to the new types of quasi-static constraints, the constraint satisfaction algorithms also have to take into account the robot kinodynamic limitations in this work. Among the robot limitations, its balanced pose, reachability and motor torques are selected particularly. The output of the algorithm is both the placement of the objects and as an addition to the previous contributions, the pose of the robot before the manipulation of the system. The validation examples were executed by us placing Golem Krang in the planner output configurations and the robot performing the necessary input. The output forces on the payload were measured and compared with the inputs to compute the gained mechanical leverage and thus, showing that our framework can be used to extend the force capabilities of robots. Additional experiments were conducted in a semi-autonomous scenario in [41] where Golem Krang performs a sequence of perception, motion planning, whole-body control, grasping and task-constrained manipulation tasks to assemble a lever-fulcrum system.

9.2 Future Work

The presented framework is based on the iterative resolution of constraint satisfaction problems where the successor problems are different from their immediate predecessors only by a small set of new variables and constraints. We propose that the information generated in the first place for the predecessor problems should be reused in the solution of successor problems. Fortunately, this notion has been adopted in the field of simultaneous localization and mapping where as an agent observes the

environment, it needs to update its map incrementally and it is advantageous to preserve the data that was generated in the previous map computation. Dellaert and Kaess [25] have proposed the incremental smoothing and mapping algorithm based on an iterative solution of the Levenberg-Marquardt algorithm with new landmark pose variables and constraints between them and the previously observed landmarks. We propose that adaptation of this framework to the functional design domain can yield significant computation gains and hence, increase the scalability of our framework.

Moreover, in future work, it would be beneficial to examine the parameterization fo the component-robot interactions that shape the design of the functional structures. In the scope of this work, we have only imposed constraints on the absolute changes in the pose of the robot. For instance, in the stacking domain we only set limits on the step sizes rather than the way the robot moves from one block to another. This approach has a number of drawbacks such as the lack of collision checks with the environment and accounting for joint limits. On the other hand, exploiting the fact that a wide range of behaviors can be represented in a lower-dimensional space, using for instance curves to model the motion trajectories, can be effective in accounting for collisions/joint limits. The motion planning literature has a number of examples where joint trajectories are represented with B-spline wavelets [87, 127]. We propose extending the parameterization of the motor interactions for functional structures with B-spline representation of behavior primitives to generate more comprehensive use behaviors and to create structures that are more reliable in real-world testing.

REFERENCES

[1] Avis, D. and Fukuda, K., "A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra," *Discrete & Computational Geometry*, vol. 8, no. 1, pp. 295–313, 1992.

[2] Azmitia, M., "Peer interaction and problem solving: When are two heads better than one?," *Child development*, pp. 87–96, 1988.

[3] Bacchus, F. and Van Run, P., "Dynamic variable ordering in csps," in *Principles and Practice of Constraint ProgrammingCP'95*, pp. 258–275, Springer, 1995.

[4] Bagnara, R., Hill, P. M., and Zaffanella, E., "The parma polyhedra library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems," *Science of Computer Programming*, vol. 72, no. 1, pp. 3–21, 2008.

[5] Bai, H., Hsu, D., Kochenderfer, M. J., and Lee, W. S., "Unmanned aircraft collision avoidance using continuous-state pomdps," *Robotics: Science and Systems VII*, vol. 1, 2012.

[6] Bejczy, A. K., "Effect of hand-based sensors on manipulator control performance," *Mechanism and Machine Theory*, vol. 12, no. 5, pp. 547–567, 1977.

[7] Bénichou, M., Gauthier, J.-M., Girodet, P., Hentges, G., Ribière, G., and Vincent, O., "Experiments in mixed-integer linear programming," *Mathematical Programming*, vol. 1, no. 1, pp. 76–94, 1971.

[8] Berenson, D., Srinivasa, S. S., and Kuffner, J., "Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, p. 0278364910396389, 2011.

[9] Berthold, T. and Gleixner, A. M., "Undercover: a primal minlp heuristic exploring a largest sub-mip," *Mathematical Programming*, vol. 144, no. 1-2, pp. 315–346, 2014.

[10] Blum, A. L. and Furst, M. L., "Fast planning through planning graph analysis," *Artificial intelligence*, vol. 90, no. 1, pp. 281–300, 1997.

[11] Boissonnat, J.-D., Sharir, M., Tagansky, B., and Yvinec, M., "Voronoi diagrams in higher dimensions under certain polyhedral distance functions," *Discrete & Computational Geometry*, vol. 19, no. 4, pp. 485–519, 1998.

[12] Brosnan, M. J., "Spatial ability in children's play with lego blocks," *Perceptual and Motor Skills*, vol. 87, no. 1, pp. 19–28, 1998.

[13] Brown, K. Q., "Voronoi diagrams from convex hulls," *Information Processing Letters*, vol. 9, no. 5, pp. 223–228, 1979.

[14] Bry, A. and Roy, N., "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 723–730, IEEE, 2011.

[15] Busemann, S., Steffen, J., and Herrmann, E., "Interactive planning of manual assembly operations: From language to motion," *Procedia CIRP*, vol. 41, pp. 224–229, 2016.

[16] Charman, P., "Solving space planning problems using constraint technology," *Nato ASI Constraint Programming: Students Presentations, TR CS*, vol. 57, no. 93, pp. 80–96, 1993.

[17] Chazelle, B., "Approximation and decomposition of shapes," *Algorithmic and Geometric Aspects of Robotics*, vol. 1, pp. 145–185, 1987.

[18] Chew, L. P., "Constrained delaunay triangulations," *Algorithmica*, vol. 4, no. 1-4, pp. 97–108, 1989.

[19] Choi, J. and Amir, E., "Combining planning and motion planning," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 238–244, IEEE, 2009.

[20] Dalbec, J. P. and Schenck, H., "On a conjecture of rose," *Journal of Pure and Applied Algebra*, vol. 165, no. 2, pp. 151–154, 2001.

[21] Dantzig, G. B., Orden, A., Wolfe, P., and others, "The generalized simplex method for minimizing a linear form under linear inequality restraints," *Pacific Journal of Mathematics*, vol. 5, no. 2, pp. 183–195, 1955.

[22] Dantzig, G. B. and Wolfe, P., "Decomposition principle for linear programs," *Operations research*, vol. 8, no. 1, pp. 101–111, 1960.

[23] Dechter, R. and Pearl, J., "Network-based heuristics for constraint-satisfaction problems," *Artificial Intelligence*, vol. 34, no. 1, pp. 1–38, 1987.

[24] Dellaert, F., "Factor graphs and gtsam: A hands-on introduction," 2012.

[25] Dellaert, F. and Kaess, M., "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[26] Dini, G. and Santochi, M., "Automated sequencing and subassembly detection in assembly planning," *CIRP Annals-Manufacturing Technology*, vol. 41, no. 1, pp. 1–4, 1992.

[27] Director, S. W. and Hachtel, G. D., "The simplicial approximation approach to design centering," *Circuits and Systems, IEEE Transactions on*, vol. 24, no. 7, pp. 363–372, 1977.

[28] Do, M. B. and Kambhampati, S., "Sapa: A multi-objective metric temporal planner," *J. Artif. Intell. Res.(JAIR)*, vol. 20, pp. 155–194, 2003.

[29] Dogar, M., Knepper, R. A., Spielberg, A., Choi, C., Christensen, H. I., and Rus, D., "Towards coordinated precision assembly with robot teams,"

[30] Donald, B., Xavier, P., Canny, J., and Reif, J., "Kinodynamic motion planning," *Journal of the ACM (JACM)*, vol. 40, no. 5, pp. 1048–1066, 1993.

[31] Dorndorf, U., P. E. and Phan-Huy, T., "A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalised precedence constraints," *Management Science*, 2000.

[32] Dornhege, C., Eyerich, P., Keller, T., Brenner, M., and Nebel, B., "Integrating task and motion planning using semantic attachments.," in *Bridging the Gap Between Task and Motion Planning*, 2010.

[33] Eastman, C. M., *Heuristic algorithms for automated space planning*. Citeseer, 1971.

[34] Elliott Fahlman, S., "A planning system for robot construction tasks," *Artificial Intelligence*, vol. 5, no. 1, pp. 1–49, 1974.

[35] Emiris, I. Z. and Fisikopoulos, V., "Efficient random-walk methods for approximating polytope volume," in *Proceedings of the thirtieth annual symposium on Computational geometry*, p. 318, ACM, 2014.

[36] Erdem, E., Haspalamutgil, K., Palaz, C., Patoglu, V., and Uras, T., "Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 4575–4581, IEEE, 2011.

[37] Erdogan, C. and Stilman, M., "Planning in constraint space: Automated design of functional structures," *ICRA*, 2013.

[38] Erdogan, C. and Stilman, M., "Autonomous realization of simple machines," in *International Symposium on Experimental Robotics (ISER)*, 2014.

[39] Erdogan, C. and Stilman, M., "Ensuring buildability of simple machine designswith task-constrained motion planning," in *Robotics, Science and System workshop - Constrained decision-making in robotics: models, algorithms, and applications*, IEEE, 2014.

[40] Erdogan, C. and Stilman, M., "Incorporating kinodynamic constraints in automated design of simple machines," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 2931–2936, IEEE, 2014.

[41] Erdogan, C. and Stilman, M., "Autonomous design of functional structures," *Advanced Robotics*, vol. 29, no. 9, pp. 625–638, 2015.

[42] Erdogan, C., Zafar, M., and Stilman, M., "Gravity and drift in force/torque measurements," 2014.

[43] Erdogan, C., Zafar, M., and Stilman, M., "Krang kinematics: A denavit-hartenberg parameterization," 2014.

[44] Eyerich, P., Keller, T., Nebel, B., and others, "Combining action and motion planning via semantic attachments," in *Proc. of Workshop on Combining Action and Motion Planning at ICAPS*, Citeseer, 2010.

[45] Floudas, C. A., Gümüs, Z. H., and Ierapetritou, M. G., "Global optimization in design under uncertainty: feasibility test and flexibility index problems," *Industrial & Engineering Chemistry Research*, vol. 40, no. 20, pp. 4267–4282, 2001.

[46] Floudas, C. A. and Lin, X., "Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review," *Computers & Chemical Engineering*, vol. 28, no. 11, pp. 2109–2129, 2004.

[47] Fox, M. S., Sadeh, N., and Baykan, C., "Constrained heuristic search," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 309–315, 1989.

[48] Geyer, P., "Component-oriented decomposition for multidisciplinary design optimization in building design," *AEI*, 2009.

[49] Goldman, R. P. and Boddy, M. S., "Expressive planning and explicit knowledge.," in *AIPS*, vol. 2, p. 3, 1996.

[50] Goldsmith, T. E. and Davenport, D. M., "Assessing structural similarity of graphs.," 1990.

[51] Goldwasser, M., Latombe, J.-C., and Motwani, R., "Complexity measures for assembly sequences," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 2, pp. 1851–1857, IEEE, 1996.

[52] Gottlob, G., Leone, N., and Scarcello, F., "Hypertree decompositions and tractable queries," in *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 21–32, ACM, 1999.

[53] Goyal, V. and Ierapetritou, M. G., "Determination of operability limits using simplicial approximation," *AIChE journal*, vol. 48, no. 12, pp. 2902–2909, 2002.

[54] Gravot, F., Cambon, S., and Alami, R., "asymov: a planner that deals with intricate symbolic and geometric problems," in *Robotics Research*, pp. 100–110, Springer, 2005.

[55] Grayson, D. R. and Stillman, M. E., "Macaulay 2, a software system for research in algebraic geometry," 2002.

[56] Grey, M., Dantam, N., Lofaro, D. M., Bobick, A., Egerstedt, M., Oh, P., and Stilman, M., "Multi-process control software for hubo2 plus robot," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, pp. 1–6, IEEE, 2013.

[57] Greytak, M. and Hover, F., "Motion planning with an analytic risk cost for holonomic vehicles," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pp. 5655–5660, IEEE, 2009.

[58] Halperin, D., Latombe, J.-C., and Wilson, R. H., "A general framework for assembly planning: The motion space approach," *Algorithmica*, vol. 26, no. 3-4, pp. 577–601, 2000.

[59] Hauser, K. and Latombe, J.-C., "Integrating task and prm motion planning: Dealing with many infeasible motion planning queries," in *ICAPS09 Workshop on Bridging the Gap between Task and Motion Planning*, 2009.

[60] Henk, M., Richter-Gebert, J., and Ziegler, G. M., "15 basic properties of convex polytopes," 1995.

[61] Hoffmann, J. and Nebel, B., "The ff planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence Research*, pp. 253–302, 2001.

[62] J. Latombe, R. W. and Cazals, F., "Assembly sequencing with tolerated parts,"

[63] Jiménez, P., Thomas, F., and Torras, C., "3d collision detection: a survey," *Computers & Graphics*, vol. 25, no. 2, pp. 269–285, 2001.

[64] Jones, D. R., Perttunen, C. D., and Stuckman, B. E., "Lipschitzian optimization without the lipschitz constant," *Journal of Optimization Theory and Applications*, vol. 79, no. 1, pp. 157–181, 1993.

[65] Jones, R. E. and Wilson, R. H., "A survey of constraints in automated assembly planning," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 2, pp. 1525–1532, IEEE, 1996.

[66] Jones, R. E., Wilson, R. H., and Calton, T. L., "Constraint-based interactive assembly planning," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 2, pp. 913–920, IEEE, 1997.

[67] Kaelbling, L. P. and Lozano-Pérez, T., "Hierarchical task and motion planning in the now," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1470–1477, IEEE, 2011.

[68] Kaelbling, L. P. and Lozano-Pérez, T., "Integrated task and motion planning in belief space," *The International Journal of Robotics Research*, p. 0278364913484072, 2013.

[69] Kaess, M., Ila, V., Roberts, R., and Dellaert, F., "The bayes tree: An algorithmic foundation for probabilistic robot mapping," in *Algorithmic Foundations of Robotics IX*, pp. 157–173, Springer, 2010.

[70] Kannan, R., Lovász, L., and Simonovits, M., "Random walks and an o*(n5) volume algorithm for convex bodies," *Random structures and algorithms*, vol. 11, no. 1, pp. 1–50, 1997.

[71] Karaman, S. and Frazzoli, E., "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[72] Kaufman, S. G., Wilson, R. H., Jones, R. E., Calton, T. L., and Ames, A. L., "The archimedes 2 mechanical assembly planning system," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 4, pp. 3361–3368, IEEE, 1996.

[73] Kavraki, L., Latombe, J.-C., and Wilson, R. H., "On the complexity of assembly partitioning," *Information Processing Letters*, vol. 48, no. 5, pp. 229–235, 1993.

[74] Kavraki, L. E., Švestka, P., Latombe, J.-C., and Overmars, M. H., "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996.

[75] Kedem, K., Livne, R., Pach, J., and Sharir, M., "On the union of jordan regions and collision-free translational motion amidst polygonal obstacles," *Discrete & Computational Geometry*, vol. 1, no. 1, pp. 59–71, 1986.

[76] Khatib, O., "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[77] Klee, V. and Minty, G. J., "How good is the simplex algorithm," tech. rep., DTIC Document, 1970.

[78] Kuffner Jr, J. and LaValle, S., "RRT-connect: An efficient approach to single-query path planning," in *ICRA*, IEEE, 2000.

[79] Lagriffoul, F., Dimitrov, D., Saffiotti, A., and Karlsson, L., "Constraint propagation on interval bounds for dealing with geometric backtracking," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 957–964, IEEE, 2012.

[80] Latecki, L. J. and Lakämper, R., "Convexity rule for shape decomposition based on discrete contour evolution," *Computer Vision and Image Understanding*, vol. 73, no. 3, pp. 441–454, 1999.

[81] Latombe, J.-C., *Robot motion planning*, vol. 124. Springer Science & Business Media, 2012.

[82] LaValle, S. M., *Planning algorithms*. Cambridge university press, 2006.

[83] LaValle, S. M. and Kuffner, J. J., "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[84] Le, D. T., Cortés, J., and Siméon, T., "A path planning approach to (dis) assembly sequencing," in *Automation Science and Engineering, 2009. CASE 2009. IEEE International Conference on*, pp. 286–291, IEEE, 2009.

[85] Lee, C. and Nigam, R., "Development of the generalized d'alembert equations of motion for mechanical manipulators," in *Decision and Control, 1983. The 22nd IEEE Conference on*, vol. 22, pp. 1205–1210, IEEE, 1983.

[86] Lee, D.-T. and Schachter, B. J., "Two algorithms for constructing a delaunay triangulation," *International Journal of Computer & Information Sciences*, vol. 9, no. 3, pp. 219–242, 1980.

[87] Lengagne, S., Mathieu, P., Kheddar, A., and Yoshida, E., "Generation of dynamic motions under continuous constraints: Efficient computation using b-splines and taylor polynomials," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 698–703, IEEE, 2010.

[88] Leuschke, G. J., "Endomorphism rings of finite global dimension," *arXiv preprint math/0505323*, 2005.

[89] Levihn, M., Scholz, J., and Stilman, M., "Hierarchical decision theoretic planning for navigation among movable obstacles," in *Algorithmic Foundations of Robotics X*, pp. 19–35, Springer, 2013.

[90] Liu, H., Liu, W., and Latecki, L. J., "Convex shape decomposition," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 97–104, IEEE, 2010.

[91] Liu, Y. and Koenig, S., "Risk-sensitive planning with one-switch utility functions: Value iteration," in *AAAI*, pp. 993–999, 2005.

[92] Liu, Y. and Koenig, S., "An exact algorithm for solving mdps under risk-sensitive planning objectives with one-switch utility functions," in *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 453–460, International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[93] Long, D. and Fox, M., "The 3rd international planning competition: Results and analysis," *J. Artif. Intell. Res.(JAIR)*, vol. 20, pp. 1–59, 2003.

[94] Lovász, L. and Deák, I., "Computational results of an o(n 4) volume algorithm," *European Journal of Operational Research*, vol. 216, no. 1, pp. 152–161, 2012.

[95] Lovász, L. and Vempala, S., "Simulated annealing in convex bodies and an o*(n4) volume algorithm," *Journal of Computer and System Sciences*, vol. 72, no. 2, pp. 392–417, 2006.

[96] Lozano-Perez, T., "Spatial planning: A configuration space approach," *Computers, IEEE Transactions on*, vol. 100, no. 2, pp. 108–120, 1983.

[97] Lozano-Pérez, T. and Kaelbling, L. P., "A constraint-based method for solving sequential manipulation planning problems," in *IROS*, 2014.

[98] Lozano-Pérez, T. and Wesley, M. A., "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[99] Lozano-Perez, T. and Winston, P. H., "Lama: a language for automatic mechanical assembly," in *Proceedings of the 5th international joint conference on Artificial intelligence*, pp. 710–716, 1977.

[100] Mackworth, A. K., "Consistency in networks of relations," *Artificial intelligence*, vol. 8, no. 1, pp. 99–118, 1977.

[101] Makhorin, A., "Gnu linear programming kit (glpk)," *Department for Applied Informatics, Moscow Aviation Institute, Moscow, Russia*, 2001.

[102] Marecki, J. and Varakantham, P., "Risk-sensitive planning in partially observable environments," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pp. 1357–1368, International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[103] Mason, M. T., "Mechanics and planning of manipulator pushing operations," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 53–71, 1986.

[104] McCarthy, J., "Situations, actions, and causal laws," tech. rep., DTIC Document, 1963.

[105] Medjdoub, B. and Yannou, B., "Separating topology and geometry in space planning," *Computer-Aided Design*, vol. 32, no. 1, pp. 39–61, 2000.

[106] Mordatch, I., Todorov, E., and Popović, Z., "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 43, 2012.

[107] Moré, J. J., "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*, pp. 105–116, Springer, 1978.

[108] Nilsson, N. J., "A mobile automaton: An application of artificial intelligence techniques," tech. rep., DTIC Document, 1969.

[109] Park, J., Haan, J., and Park, F. C., "Convex optimization algorithms for active balancing of humanoid robots," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 817–822, 2007.

[110] Perny, P., Spanjaard, O., and Storme, L.-X., "State space search for risk-averse agents.," in *IJCAI*, pp. 2353–2358, 2007.

[111] Plaku, E. and Hager, G. D., "Sampling-based motion and symbolic action planning with geometric and differential constraints," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 5002–5008, IEEE, 2010.

[112] Pollack, R., Sharir, M., and Sifrony, S., "Separating two simple polygons by a sequence of translations," *Discrete & Computational Geometry*, vol. 3, no. 1, pp. 123–136, 1988.

[113] Powell, M. J., "A direct search optimization method that models the objective and constraint functions by linear interpolation," *Advances in optimization and numerical analysis*, pp. 51–67, 1994.

[114] Raymond, J. W., Gardiner, E. J., and Willett, P., "Rascal: Calculation of graph similarity using maximum common edge subgraphs," *The Computer Journal*, vol. 45, no. 6, pp. 631–644, 2002.

[115] Runarsson, T. P. and Yao, X., "Search biases in constrained evolutionary optimization," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 35, no. 2, pp. 233–243, 2005.

[116] Russell, S. and Norvig, P., *Artificial intelligence: A Modern Approach*. Prentice Hall, 2010.

[117] Seo, J., Yim, M., and Kumar, V., "Assembly planning for planar structures of a brick wall pattern with rectangular modular robots," in *Automation Science and Engineering (CASE), 2013 IEEE International Conference on*, pp. 1016–1021, IEEE, 2013.

[118] Shewchuk, J. R., "A condition guaranteeing the existence of higher-dimensional constrained delaunay triangulations," in *Proceedings of the fourteenth annual symposium on Computational geometry*, pp. 76–85, ACM, 1998.

[119] Srivastava, S., Riano, L., Russell, S., and Abbeel, P., "Using classical planners for tasks with continuous operators in robotics," in *Intl. Conf. on Automated Planning and Scheduling*, 2013.

[120] Stilman, M., "Task constrained motion planning in robot joint space," in *IROS*, IEEE, 2007.

[121] Stilman, M., *Navigation among movable obstacles*. PhD thesis, 2007.

[122] Stilman, M., Olson, J., and Gloss, W., "Golem krang: Dynamically stable humanoid robot for mobile manipulation," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 3304–3309, IEEE, 2010.

[123] Stone, P., Veloso, M. M., and Blythe, J., "The need for different domain-independent heuristics.," in *AIPS*, pp. 164–169, 1994.

[124] Sucan, I. A. and Kavraki, E., "Mobile manipulation: Encoding motion planning options using task motion multigraphs," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 5492–5498, IEEE, 2011.

[125] Thrun, S., Langford, J., and Verma, V., "Risk sensitive particle filters," *Advances in neural information processing systems*, vol. 2, pp. 961–968, 2002.

[126] Tolani, D., Goswami, A., and Badler, N., "Real-time inverse kinematics techniques for anthropomorphic limbs," *Graphical models*, 2000.

[127] Ude, A., Atkeson, C. G., and Riley, M., "Planning of joint trajectories for humanoid robots using b-spline wavelets," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 3, pp. 2223–2228, IEEE, 2000.

[128] Ueda, K. and Yamashita, N., "On a global complexity bound of the levenberg-marquardt method," *Journal of optimization theory and applications*, vol. 147, no. 3, pp. 443–453, 2010.

[129] Vachhani, L., Mahindrakar, A. D., and Sridharan, K., "Mobile robot navigation through a hardware-efficient implementation for control-law-based construction of generalized voronoi diagram," *Mechatronics, IEEE/ASME Transactions on*, vol. 16, no. 6, pp. 1083–1095, 2011.

[130] Veltkamp, R. C., "Shape matching: similarity measures and algorithms," in *Shape Modeling and Applications, SMI 2001 International Conference on.*, pp. 188–197, IEEE, 2001.

[131] Vidal, V. and Geffner, H., "Branching and pruning: An optimal temporal pocl planner based on constraint programming," *AI*, 2006.

[132] Vilain, M. B. and Kautz, H. A., "Constraint propagation algorithms for temporal reasoning.," in *Aaai*, vol. 86, pp. 377–382, 1986.

[133] Walker, R. J., "An enumerative technique for a class of combinatorial problems," in *Proceedings of Symposia in Applied Mathematics*, vol. 10, pp. 91–94, American Math. Soc Providence, RI, 1960.

[134] Watson, D. F., "Computing the n-dimensional delaunay tessellation with application to voronoi polytopes," *The computer journal*, vol. 24, no. 2, pp. 167–172, 1981.

[135] Whitney, D. E., "Resolved motion rate control of manipulators and human prostheses.," *IEEE Transactions on man-machine systems*, 1969.

[136] Wilson, R. H., *On geometric assembly planning.* PhD thesis, stanford university, 1992.

[137] Wilson, R. H., Kavraki, L., Latombe, J.-C., and Lozano-Pérez, T., "Two-handed assembly sequencing," *The International journal of robotics research*, vol. 14, no. 4, pp. 335–350, 1995.

[138] Wilson, R. H. and Latombe, J.-C., "Geometric reasoning about mechanical assembly," *Artificial Intelligence*, vol. 71, no. 2, pp. 371–396, 1994.

[139] Wolfe, J., Marthi, B., and Russell, S. J., "Combined task and motion planning for mobile manipulation.," in *ICAPS*, pp. 254–258, 2010.

[140] Xu, L. D., Wang, C., Bi, Z., and Yu, J., "Object-oriented templates for automated assembly planning of complex products," *Automation Science and Engineering, IEEE Transactions on*, vol. 11, no. 2, pp. 492–503, 2014.

[141] Yu, L., Y. S. T. C. T. D. C. T. and Osher, S., "Make it home: automatic optim. of furniture arrangement.," *Siggraph*, 2011.

[142] Zafar, M., Erdogan, C., and Stilman, M., "Krang: Center of mass estimation," 2014.

[143] Zafar, M., Erdogan, C., and Stilman, M., "Towards stable balancing," 2014.

[144] Zager, L. A. and Verghese, G. C., "Graph similarity scoring and matching," *Applied mathematics letters*, vol. 21, no. 1, pp. 86–94, 2008.